

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 готельного комплексу

Студента (ки) курсу 2 групи ІТ-01
спеціальності 121 «Інженерія програмного
забезпечення»

Колесника Романа

(прізвище та ініціали)

Керівник _____

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

Національна шкала

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

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

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

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

Київ – 2021 рік

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА	5
1.1 Опис предметної області	5
1.2 Опис вхідних даних	5
1.3 Опис вихідних даних	6
1.4 Інфологічна модель бази даних	9
1.5 Опис сутностей	9
1.6 Опис атрибутів	10
1.7 Опис зв'язків	12
1.8 ER-діаграма	14
1.9 Збереження цілісності бази даних	14
РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА	20
2.1 Створення бази даних за допомогою MySQL Server	20
2.2 Створення таблиць	20
2.3 Створення діаграми	24
2.4 Заповнення даними таблиць бази даних	24
2.5 Створення користувачів для доступу бази даних на різних рівнях	28
2.6 Створення збережених процедур	29
2.7 Створення тригерів	30
2.8 Створення представлень	31
2.9 Створення функцій	33
РОЗДІЛ 3. DML-ЗАПИТИ	35
3.1 Організація вибірки інформації з бази даних	35
3.2 Проста вибірка даних	35
3.3 Вибірка з діапазону та множини	38
3.4 Вибірка з регулярними виразами	39
3.5 Вибірка із запитом зменшення піль	40
3.6 Вибірка із групуванням	41
3.7 Вибірка із впорядкуванням	42
3.8 Вибірка із комбінацією	43
3.9 Вибірка inner select та union	44
ВИСНОВКИ	46
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	47
ДОДАТКИ	48

ВСТУП

База даних (БД) — це організована структура, призначена для зберігання, зміни й обробки взаємопов'язаної інформації, переважно великих обсягів. Бази даних активно використовують для динамічних сайтів зі значними обсягами даних — часто це інтернет-магазини, портали, корпоративні сайти.

Такі сайти зазвичай розроблені за допомогою серверної мови програмування (наприклад, PHP) або на базі CMS (наприклад, WordPress), і не мають готових сторінок з даними за аналогією з HTML-сайтами. Сторінки динамічних сайтів формуються «на льоту» в результаті взаємодії скриптів і баз даних після відповідного запиту клієнта до веб-сервера [1].

Реляційна база даних - це набір даних зі зв'язками між ними. Ці дані організовані у вигляді набору таблиць, що складаються із стовпців і рядків. У таблицях зберігається інформація про об'єкти, представлені в базі даних. У кожному стовпці таблиці зберігається певний тип даних, в кожному осередку - значення атрибута.

Кожен рядок таблиці представляє собою набір пов'язаних значень, що відносяться до одного об'єкту або сутності. Кожен рядок в таблиці може бути позначена унікальним ідентифікатором, званим первинним ключем, а рядки з декількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів.

Система управління базами даних (СУБД) - це програмна прошарок між користувачем і сервером. Тому вона дозволяє абстрагувати користувача від системного бачення БД, а системі надає спосіб взаємодіяти з користувачем. [2]

Таким чином, було проаналізовано кілька систем управління та обрано одну з них (див. порівняння додаток Б).

MySQL - це система управління базами даних, яка використовується для підтримки реляційних баз даних. Це програмне забезпечення з відкритим кодом, що підтримується корпорацією Oracle. Спочатку вона була заснована шведською компанією під назвою MYSQL AB, яка згодом була придбана соня-

чними мікросистемами і, нарешті, є корпорацією Oracle. Оскільки це система баз даних з відкритим кодом, вихідний код можна змінювати відповідно до наших потреб [3].

У рамках цієї курсової роботи розроблено реляційну базу даних “База даних готельного комплексу” за допомогою середовища управління базами даних MySQL Microsoft Server.

Для цього поставлені наступні завдання:

- проаналізувати предметне середовище, визначати сутності та атрибути, зв'язки між об'єктами;
- побудувати ER-модель заданого предметного середовища;
- побудувати реляційну схему бази даних на основі заданої ER-моделі
- розробити відповідні скрипти з використанням засобів мови SQL для побудови спроектованої бази даних;
- імпортувати дані в розроблену базу даних;
- виконати запити до розробленої бази даних.

Для зручного доступу та розробки за допомогою високотехнологічного графічного інтерфейсу користувача, що адаптовується під нього, використано крос-платформове середовище розробки — DataGrip (див. додаток А).

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Опис предметної області

Інформаційна система відповідає за забезпечення інформаційних процесів, забезпечення створення, поширення, використання, збереження і знищення інформації. Сама інформаційна база складається з однієї або декількох баз даних.

При розробці реляційної бази даних «База даних готельного комплексу» було проведено дослідження предметної області.

Таким чином, наявні готелі, що повинні обслуговувати клієнтів. У готелях працюють люди з різними посадами. Причому у різних готелях різний склад працівників. Є доступ до даних сутностей, зокрема до адрес. Отже, такі поля, зокрема адреси, варто інкапсулювати, тобто відокремити.

Отже, необхідно збирати інформацію від клієнтів про їхні замовлення та обслуговувати згідно з ціновим прайсом певного готелю та своєчасно надавати інформацію про послуги.

1.2 Опис вхідних даних

Проаналізувавши предметну область, необхідно наповнити базу даних значеннями.

При розробці реляційної бази даних «База даних готельного комплексу» були виділені наступні вхідні дані:

- інформація про готель
- інформація про клієнта-замовника
- інформація про працівників
- інформація про послуги готелю
- інформація про замовлення

1.3 Опис вихідних даних

Вихідні дані - повідомлення і результати, які видаються самою внаслідок виклику запитів та зберігаються в таблицях [5].

Таблиця 1.3.1 – comfort (тип кімнати в готелі)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
description	varchar	Опис комфорту кімнати

Таблиця 1.3.2 – room (кімната в готелі)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
comfortID	int	Вказує на унікальний ідентифікатор комфорту

Таблиця 1.3.3 – order (замовлення клієнта)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
clientID	int	Вказує на унікальний ідентифікатор клієнта
payment	int	Ціна за послугу
hotelID	int	Вказує на унікальний ідентифікатор готелю
roomID	int	Вказує на унікальний ідентифікатор кімнати
inDate	date	Дата заселення
livingDays	int	Тривалість проживання у днях

Таблиця 1.3.4 – client (клієнт)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
name	varchar	Ім'я
surname	varchar	Прізвище
addressID	int	Вказує на унікальний ідентифікатор адреси

Таблиця 1.3.5 – address (адреса)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
description	varchar	Повна адреса

Таблиця 1.3.6 – hotel (готель)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
name	varchar	Назва готелю
dateInstallation	date	Дата спорудження
cityID	int	Вказує на унікальний ідентифікатор міста
addressID	int	Вказує на унікальний ідентифікатор адреси

Таблиця 1.3.7 – employee (працівник)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
name	varchar	Ім'я

surname	varchar	Прізвище
addressID	int	Вказує на унікальний ідентифікатор адреси
positionID	int	Вказує на унікальний ідентифікатор посади
hotelID	int	Вказує на унікальний ідентифікатор готелю

Таблиця 1.3.8 – position (посада працівника)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
description	varchar	Повна посада

Таблиця 1.3.9 – city (місто)

Назва поля	Тип поля	Опис поля
id	int	Унікальний ідентифікатор
name	varchar	Ім'я
population	int	Населення
countryID	int	Вказує на унікальний ідентифікатор країни

Таблиця 1.3.10 – country (країна)

Назва поля	Тип поля	Опис поля
------------	----------	-----------

id	int	Унікальний ідентифікатор
name	varchar	Ім'я
population	int	Населення

1.4 Інфологічна модель бази даних

Мета інфологічного моделювання - забезпечення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створюваній базі даних. Тому інфологічну модель даних намагаються будувати по аналогії з природною мовою (останній не може бути використаний в чистому вигляді із-за складності комп'ютерної обробки текстів і неоднозначності будь-якої природної мови). Основними конструктивними елементами інфологічних моделей є суті, зв'язки між ними і їх властивості (атрибути).

Суть - будь-який помітний об'єкт (об'єкт, який ми можемо відрізнити від іншого), інформацію про який необхідно зберігати в базі даних. Суттю можуть бути люди, місця, літаки, рейси, смак, колір і т.д. Необхідно розрізняти такі поняття, як тип суті і екземпляр суті. Поняття тип суті відноситься до набору однорідних осіб, предметів, подій або ідей, виступаючих як ціле. Екземпляр суті відноситься до конкретної речі в наборі. Наприклад, типом суті може бути місто, а екземпляром - Мінськ, Київ і т.д [6].

1.5 Опис сутностей

Одним із елементом інфологічної моделі "сутність-зв'язок" є сутності. Сутність – це те, про що накопичується інформація в інформаційній системі і що може бути однозначно унікальне ідентифіковане.

При цьому ім'я сутності повинно відображати клас об'єкта або тип об'єкта. Наприклад, сутністю є людина, де її атрибути — це дані про цю людину.

У відповідності з описом предметної області були отримано такі сутності:

- comfort — містить інформацію про комфорт кімнати в готелі

- room — містить інформацію про кімнати в готелі
- order — містить інформацію про замовлення
- client — містить інформацію про клієнта-замовника
- address — містить інформацію про усі адреси
- hotel — містить інформацію про готель
- city — містить інформацію про місто
- employee — містить інформацію про працівників готелю
- position — містить інформацію про посаду працівників
- country — містить інформацію про країну

1.6 Опис атрибутів

Атрибут, або поле, — елемент даних у реляційних базах даних. Наприклад, для сутності людина атрибутами будуть дата народження, маса, ріст, прізвище тощо. Для кожного атрибута є безліч значень.

У відповідності з описом предметної області були виділені наступні атрибути у кожній сутності:

1) таблиця comfort містить:

- id int Унікальний ідентифікатор
- description varchar Опис комфорту кімнати

2) таблиця room містить:

- id int Унікальний ідентифікатор
- comfortID int Вказує на унікальний ідентифікатор комфорту

3) таблиця order

- id int Унікальний ідентифікатор
- clientID int Вказує на унікальний ідентифікатор клієнта
- payment int Ціна за послугу
- hotelID int Вказує на унікальний ідентифікатор готелю

- roomID int Вказує на унікальний ідентифікатор кімнати
- inDate date Дата заселення
- livingDays int Тривалість проживання у днях

4) таблиця client містить:

- id int Унікальний ідентифікатор
- name varchar Ім'я
- surname varchar Прізвище
- addressID int Вказує на унікальний ідентифікатор адреси

5) таблиця address містить:

- id int Унікальний ідентифікатор
- description varchar Повна адреса

6) таблиця hotel містить:

- id int Унікальний ідентифікатор
- name varchar Назва готелю
- dateInstallation date Дата спорудження
- cityID int Вказує на унікальний ідентифікатор міста
- addressID int Вказує на унікальний ідентифікатор адреси

7) таблиця employee містить:

- id int Унікальний ідентифікатор
- name varchar Ім'я
- surname varchar Прізвище
- addressID int Вказує на унікальний ідентифікатор адреси

8) таблиця position містить:

- id int Унікальний ідентифікатор

- description varchar Повна посада

9) таблиця city містить:

- id int Унікальний ідентифікатор
- name varchar Ім'я
- population int Населення
- countryID int Вказує на унікальний ідентифікатор країни

10) таблиця country містить:

- id int Унікальний ідентифікатор
- name varchar Ім'я
- population int Населення

1.7 Опис зв'язків

Дві сутності можуть пов'язуватися через зв'язок екземплярів однієї сутності з екземплярами іншої сутності. Основна вимога бази даних - вміти знаходити одну сутність за значеннями інших, для чого необхідно встановити зв'язок між ними.

Так як часто в базах даних створюють більше 50 сутностей, то між цими сутностями може бути дуже багато зв'язків. Складність інфологічної моделі визначається наявністю великої кількості зв'язків.

Між таблицями можуть бути встановлені зв'язки таких типів: один-до-одного, один-до-багатьох або багато-до-багатьох.

Зв'язок один-до-одного (1:1) виявляє себе, коли одному значенню поля однієї таблиці відповідає єдине значення поля другої таблиці та, навпаки, одному значенню поля другої таблиці — єдине значення поля першої.

Зв'язок один-до-багатьох (1:n) має місце, коли одному значенню поля першої таблиці може відповідати декілька значень поля другої таблиці, а кожному значенню поля другої таблиці — тільки єдине значення поля першої.

Зв'язок багато-до-багатьох (n:n) має місце, коли кожному значенню поля першої таблиці відповідає декілька значень поля другої таблиці й кожному значенню другої таблиці відповідає декілька значень першої таблиці [8, 9].

Таблиця 1.7.1 – Зв'язки реляційної бази даних

Батьківська таблиця	Зв'язок	Дочірня таблиця
room	n:n	comfort
order	n:n	room
order	n:n	client
client	1:1	address
hotel	1:1	address
order	n:1	hotel
employee	n:1	hotel
employee	1:1	address
employee	1:1	position
hotel	n:1	city
city	n:1	country

1.8 ER-діаграма

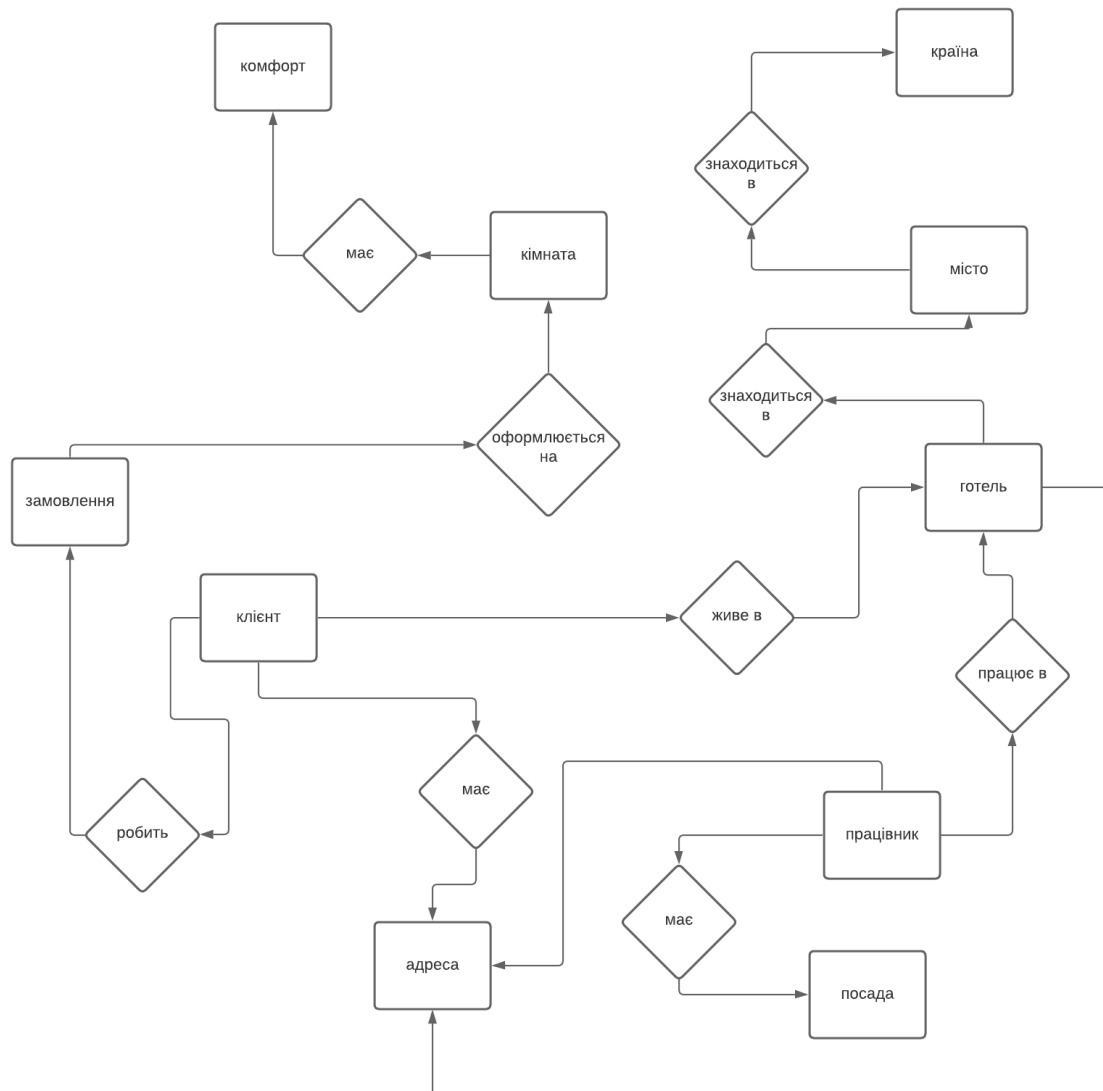


Рис. 1.8.1 – діаграма зв'язок-сутність

1.9 Збереження цілісності бази даних

Інфологічні моделі даних використовуються на ранніх стадіях проектування для опису структур даних у процесі розробки додатка, а даталогічні моделі вже підтримуються конкретними СУБД. Для моделювання даталогічної моделі ми використаємо програму AllFusion ERwin Data Modeler (ERwin) Даталогічна

модель є моделлю логічного рівня і являє собою відображення логічних зв'язків між елементами даних безвідносно до їхнього змісту й середовищу зберігання.

Ця модель будується в термінах інформаційних одиниць, припустимих у тієї конкретної СУБД, у середовищі якої ми проектуємо базу даних. Етап створення даталогічної моделі називається даталогічним проектуванням. Опис логічної структури бази даних мовою СУБД називається схемою.

Хоча даталогічне проектування є логічною структурою бази даних, на нього впливають можливості фізичної організації даних, що представляються конкретної СУБД. Тому знання особливостей фізичної організації даних є корисним при проектуванні логічної структури. Логічна структура бази даних, а також сама заповнена даними база даних є відображенням реальної предметної області. Тому на вибір проектних розв'язків найбезпосередніший вплив виявляє специфіка відображуваної предметної області, відбита в інфологічній моделі [10].

Критично важливим є дотримання умов виконання, для чого кожному полю встановлені певні обмеження і тип поля.

Таблиця 1.9.1 — обмеження таблиці comfort

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
description	Опис комфорту кімнати	varchar	32	Not null

Таблиця 1.9.2 — обмеження таблиці room

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-

comfortID	Вказує на унікальний ідентифікатор комфорту	int	4	Not null
-----------	---	-----	---	----------

Таблиця 1.9.3 — обмеження таблиці order

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
clientID	Вказує на унікальний ідентифікатор комфорту	int	4	Not null
payment	Ціна за послугу	int	4	Not null
hotelID	Вказує на унікальний ідентифікатор готелю	int	4	Not null
roomID	Вказує на унікальний ідентифікатор кімнати	int	4	Not null
inDate	Дата заселення	date	4	Not null
livingDays	Тривалість проживання у днях	int	4	Not null

Таблиця 1.9.4 — обмеження таблиці client

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
name	Ім'я	varchar	32	-
surname	Прізвище	varchar	32	Not null
addressID	Вказує на унікальний ідентифікатор адреси	int	4	Not null

Таблиця 1.9.5 — обмеження таблиці address

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
description	Повна адреса	varchar	32	-

Таблиця 1.9.6 — обмеження таблиці hotel

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
name	Повна адреса	varchar	32	-
dateInstallation	Дата спорудження	date	4	-
cityID	Вказує на унікальний ідентифікатор міста	int	4	Not null
addressID	Вказує на унікальний	int	4	Not null

	ідентифікатор адреси			
--	-------------------------	--	--	--

Таблиця 1.9.7 — обмеження таблиці employee

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
name	Ім'я	varchar	32	-
surname	Прізвище	varchar	32	Not null
addressID	Вказує на унікальний ідентифікатор адреси	int	4	Not null
positionID	Вказує на унікальний ідентифікатор посади	int	4	Not null
hotelID	Вказує на унікальний ідентифікатор готелю	int	4	Not null

Таблиця 1.9.8 — обмеження таблиці position

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
description	Повна посада	varchar	32	Not null

Таблиця 1.9.9 — обмеження таблиці city

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
name	Ім'я	varchar	32	Not null
population	Населення	int	4	-
countryID	Вказує на унікальний ідентифікатор країни	int	4	Not null

Таблиця 1.9.10 — обмеження таблиці country

Назва атрибуту	Опис атрибуту	Тип поля	Розмір поля	Обмеження
id	Унікальний ідентифікатор	int	4	-
name	Ім'я	varchar	32	Not null
population	Населення	int	4	-

РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА

2.1 Створення бази даних за допомогою MySQL Server

Для роботи на Linux Ubuntu в середовищі потрібно виконати наступні кроки:

- інсталювання MySQL Server `$sudo apt install mysql-server`
- інсталювати DataGrip `$sudo tar -xzf jetbrains-toolbox-1.17.7391.tar.gz -C /opt`
- налаштувати сервер `$sudo mysql_secure_installation`

`$sudo mysql`

`$SELECT user,authentication_string,plugin,host FROM mysql.user ;`

`$ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'YOUR_PASSWORD';`

`$FLUSH PRIVILEGES ;`

- створити підключення до локального сервера та увійти зі створеного користувача (див. рис. 2.1.1)

- перейти в консоль і створити нову схему `CREATE SCHEMA hotelComplex`

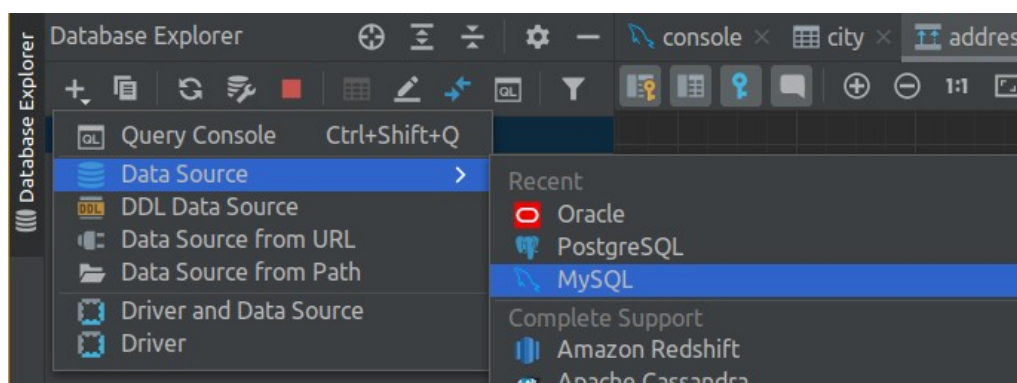


Рис. 2.1.1 — Створення підключення до локального сервера

2.2 Створення таблиць

Необхідним критерієм є створення таблиць, це виконано за допомогою наступних команд у консолі:

```
create table hotelComplex.address
(
  id          int auto_increment
  primary key,
  description varchar(32) not null
```

```

);

create table hotelComplex.client
(
    id      int auto_increment
        primary key,
    name    varchar(32) null,
    surname varchar(32) not null,
    addressID int      not null,
    constraint client_address_id_fk
        foreign key (addressID) references hotelComplex.address (id)
);

create table hotelComplex.comfort
(
    id      int auto_increment
        primary key,
    description varchar(128) not null
);

create table hotelComplex.country
(
    id      int auto_increment
        primary key,
    name    varchar(32) not null,
    population int      null
);

create table hotelComplex.city
(
    id      int auto_increment
        primary key,
    name    varchar(32) not null,
    population int      null,
    countryID int      not null,
    constraint city_country_id_fk
        foreign key (countryID) references hotelComplex.country (id)
);

create table hotelComplex.hotel

```

```
(
    id          int auto_increment
        primary key,
    name        varchar(32) null,
    dateInstallation date    null,
    cityID      int          not null,
    addressID   int          not null,
    constraint hotel_address_id_fk
        foreign key (addressID) references hotelComplex.address (id),
    constraint hotel_city_id_fk
        foreign key (cityID) references hotelComplex.city (id)
);
```

```
create table hotelComplex.position
```

```
(
    id          int auto_increment
        primary key,
    description varchar(128) not null
);
```

```
create table hotelComplex.employee
```

```
(
    id          int auto_increment
        primary key,
    name        varchar(32) null,
    surname     varchar(32) not null,
    addressID   int          not null,
    positionID  int          not null,
    hotelID     int          not null,
    constraint employee_address_id_fk
        foreign key (addressID) references hotelComplex.address (id),
    constraint employee_hotel_id_fk
        foreign key (hotelID) references hotelComplex.hotel (id),
    constraint employee_position_id_fk
        foreign key (positionID) references hotelComplex.position (id)
);
```

```
create table hotelComplex.room
```

```
(
    id          int auto_increment
```

```

        primary key,
comfortID int not null,
constraint room_comfort_id_fk
        foreign key (comfortID) references hotelComplex.comfort (id)
);

create table hotelComplex.`order`
(
    id      int auto_increment
        primary key,
    clientID int not null,
    payment int not null,
    hotelID  int not null,
    roomID   int not null,
    inDate   date not null,
    livingDays int not null,
    constraint order_client_id_fk
        foreign key (clientID) references hotelComplex.client (id),
    constraint order_room_id_fk
        foreign key (roomID) references hotelComplex.room (id),
    constraint orders_hotel_id_fk
        foreign key (hotelID) references hotelComplex.hotel (id)
);

```

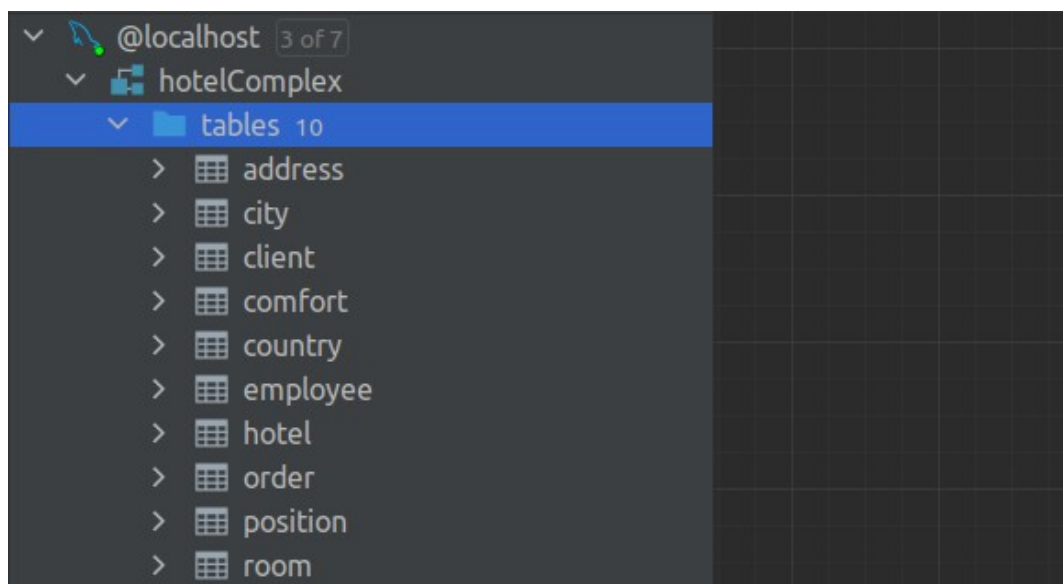


Рис. 2.2.1 — створені таблиці відображаються у схемі hotelComplex

2.3 Створення діаграми

Діаграму може бути згенеровано за допомогою Diagram DataGrip generator. Для цього потрібно натиснути ПКМ по схемі та обрати “згенерувати діаграму” у розділі діаграми або натиснути гарячу клавішу Ctrl + Shift + U.

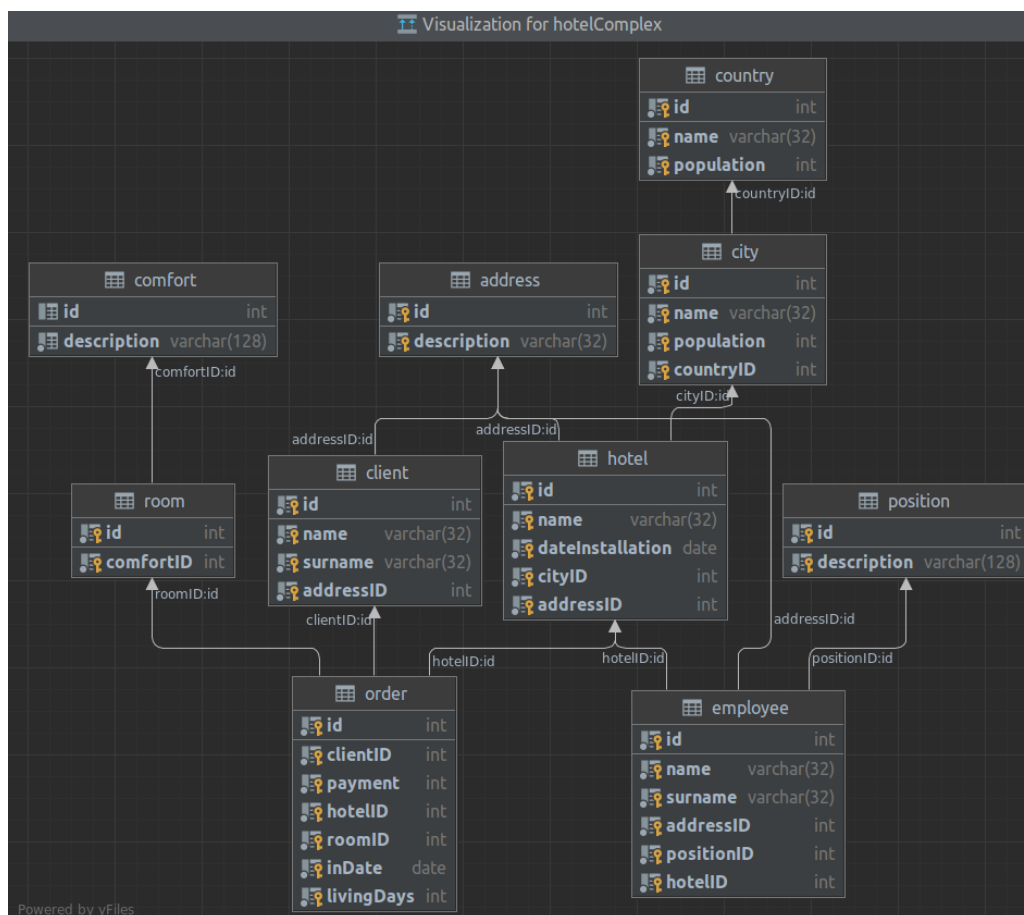


Рис. 2.3.1 — генерована діаграма таблиць схеми hotelComplex

2.4 Заповнення даними таблиць бази даних

За допомогою sql заповнюємо таблиці даними:

```
INSERT INTO hotelComplex.address (id, description) VALUES (1, 'Lesi Ukrainku 30');
INSERT INTO hotelComplex.address (id, description) VALUES (2, 'Stepand Banderu 301');
INSERT INTO hotelComplex.address (id, description) VALUES (3, 'Josepa Slipogo 1');
INSERT INTO hotelComplex.address (id, description) VALUES (4, 'Gorduza Olexindra 69');
INSERT INTO hotelComplex.address (id, description) VALUES (5, 'Ivana Poluia 90');
```



```

INSERT INTO hotelComplex.address (id, description) VALUES (6, 'Chervonoi Ruty 449');
INSERT INTO hotelComplex.address (id, description) VALUES (7, 'Poplavskogo 24');
INSERT INTO hotelComplex.address (id, description) VALUES (8, 'Chaldayevea 99');
INSERT INTO hotelComplex.address (id, description) VALUES (9, 'Ivana Franka 327');
INSERT INTO hotelComplex.address (id, description) VALUES (10, 'Billy Herrington 2 ');

```

```

INSERT INTO hotelComplex.comfort (id, description) VALUES (1, 'lux');
INSERT INTO hotelComplex.comfort (id, description) VALUES (2, 'norm');
INSERT INTO hotelComplex.comfort (id, description) VALUES (3, '3/10');
INSERT INTO hotelComplex.comfort (id, description) VALUES (4, 'classic');
INSERT INTO hotelComplex.comfort (id, description) VALUES (5, 'half-lux');
INSERT INTO hotelComplex.comfort (id, description) VALUES (6, 'double');
INSERT INTO hotelComplex.comfort (id, description) VALUES (7, 'triple');
INSERT INTO hotelComplex.comfort (id, description) VALUES (8, 'highlevel');

```

```

INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (1, 'ZZBYwPGbAF',
'nIFGdQcjdb', 1);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (2, 'yACKfjvDqm',
'qpDIwyDKus', 2);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (3, 'SaOgDzMUxh',
'APUFRxIRBd', 3);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (4, 'GJRhjhPhVx',
'RggZrYfakd', 4);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (5, 'osWhiuwGsv',
'iFSFnPGdcD', 5);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (6, 'IHOFkKKOMN',
'rxVISxoTzO', 6);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (7, 'nWzJZvBSvD',
'UmZyWmObbF', 7);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (8, 'KsKsfVUpOM',
'kBXfnDyURF', 8);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (9, 'EFBNzUwIYr',
'bTQDtwPdDf', 1);
INSERT INTO hotelComplex.client (id, name, surname, addressID) VALUES (10, 'iHguohEIaH',
'sBGKHbbBxJ', 7);

```

```

INSERT INTO hotelComplex.room (id, comfortID) VALUES (1, 1);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (2, 2);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (3, 3);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (4, 4);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (5, 5);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (6, 6);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (7, 7);
INSERT INTO hotelComplex.room (id, comfortID) VALUES (8, 8);

```

```

INSERT INTO hotelComplex.city (id, name, population, countryID) VALUES (1, 'Ternopil', null, 1);
INSERT INTO hotelComplex.city (id, name, population, countryID) VALUES (2, 'Lviv', null, 1);
INSERT INTO hotelComplex.city (id, name, population, countryID) VALUES (3, 'Kyiv', null, 1);
INSERT INTO hotelComplex.city (id, name, population, countryID) VALUES (4, 'Dobryanka', null, 1);
INSERT INTO hotelComplex.city (id, name, population, countryID) VALUES (5, 'Mykolaiv', null, 1);

```

```

INSERT INTO hotelComplex.country (id, name, population) VALUES (1, 'Ukraine', 42000000);
INSERT INTO hotelComplex.country (id, name, population) VALUES (2, 'Germany', null);
INSERT INTO hotelComplex.country (id, name, population) VALUES (3, 'USA', null);
INSERT INTO hotelComplex.country (id, name, population) VALUES (4, 'Netherlands', null);

```

```

INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (1, 'alexuk', 'yana', 2, 1, 1);
INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (2, 'olena', 'golovach', 2, 6, 1);
INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (3, 'nick', 'vuichich', 3, 7, 1);
INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (4, null, 'gorbatui', 4, 4, 1);
INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (5, null, 'chorna', 5, 9, 1);

```

```
INSERT INTO hotelComplex.employee (id, name, surname, addressID, positionID, hotelID)
VALUES (6, 'habib', 'nasran', 9, 1, 1);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (1,
'Hilton', '2009-07-06', 4, 1);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (2,
'KyivHotel', '2011-01-31', 4, 2);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (3,
'Ritz', '2011-03-17', 4, 3);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (4,
'Knife', '2013-02-25', 4, 4);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (5,
'ElitBarbarian', '2014-07-01', 4, 5);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (6,
'AnimePlanet', '2014-11-10', 4, 6);
```

```
INSERT INTO hotelComplex.hotel (id, name, dateInstallation, cityID, addressID) VALUES (7,
'Flamingo', '2017-08-09', 4, 7);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (1, 1, 800, 1, 1, '2012-09-20', 8);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (2, 2, 1600, 1, 2, '2013-02-06', 15);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (3, 3, 33000, 1, 3, '2015-07-24', 30);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (4, 4, 1900, 1, 4, '2019-02-06', 40);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (5, 5, 15, 1, 5, '2011-10-12', 15);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (6, 6, 1747, 1, 6, '2012-10-03', 1);
```

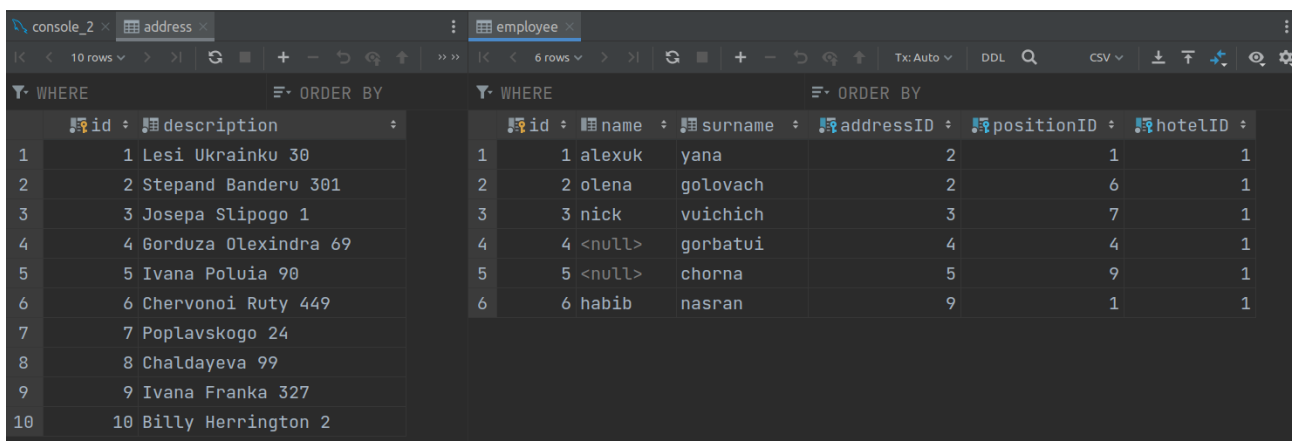
```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (7, 7, 100, 1, 7, '2013-01-24', 0);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (8, 8, 608, 1, 8, '2013-12-10', 9);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (9, 9, 100, 6, 2, '2015-07-24', 2);
```

```
INSERT INTO hotelComplex.orders (id, clientID, payment, hotelID, roomID, inDate, livingDays)
VALUES (10, 10, 300, 7, 3, '2019-02-06', 3);
```

```
INSERT INTO hotelComplex.position (id, description) VALUES (1, 'shaurma seller');
INSERT INTO hotelComplex.position (id, description) VALUES (2, 'florist');
INSERT INTO hotelComplex.position (id, description) VALUES (3, 'reception');
INSERT INTO hotelComplex.position (id, description) VALUES (4, 'concierge');
INSERT INTO hotelComplex.position (id, description) VALUES (5, 'barman');
INSERT INTO hotelComplex.position (id, description) VALUES (6, 'cleaner');
INSERT INTO hotelComplex.position (id, description) VALUES (7, 'driver');
INSERT INTO hotelComplex.position (id, description) VALUES (8, 'manager');
INSERT INTO hotelComplex.position (id, description) VALUES (9, 'director');
```



The screenshot shows a database console with two tables displayed. The 'address' table has 10 rows, and the 'employee' table has 6 rows. The 'employee' table includes columns for id, name, surname, addressID, positionID, and hotelID.

id	description
1	1 Lesi Ukraïku 30
2	2 Stepand Banderu 301
3	3 Josepa Slipogo 1
4	4 Gorduza Olexindra 69
5	5 Ivana Poluia 90
6	6 Chervonoi Ruty 449
7	7 Poplavskogo 24
8	8 Chaldayeva 99
9	9 Ivana Franka 327
10	10 Billy Herrington 2

id	name	surname	addressID	positionID	hotelID
1	alexuk	yana	2	1	1
2	olena	golovach	2	6	1
3	nick	vuichich	3	7	1
4	<null>	gorbatui	4	4	1
5	<null>	chorna	5	9	1
6	habib	nasran	9	1	1

Рис 2.4.1 — Таблиці address та employee після виконання запиту на insert

2.5 Створення користувачів для доступу бази даних на різних рівнях

Наведено приклад доступу до бази даних новим користувачам: databaseTester та anon. Їм надано усі права, що і root. Спершу ініціалізовано користувачів із заданим логіном та паролем, а потім їм надано права, перезапущено загальні привілеї.

```
14 CREATE USER 'databaseTester'@'localhost' IDENTIFIED BY 'password';
15 GRANT ALL PRIVILEGES ON * . * TO 'databaseTester'@'localhost';
16 FLUSH PRIVILEGES;
17 SHOW GRANTS FOR 'databaseTester'@'localhost';
18 # mysql -u databaseTester -p
19
20
21 ✓ CREATE USER 'anon'@'localhost' IDENTIFIED BY 'qwerty';
22 ✓ GRANT ALL PRIVILEGES ON * . * TO 'anon'@'localhost';
23 ✓ FLUSH PRIVILEGES;
24 ✓ SHOW GRANTS FOR 'anon'@'localhost';
25 # mysql -u databaseTester -p
```

Output Result 6-4 X

2 rows

Grants for anon@localhost

1	GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ...
2	GRANT APPLICATION_PASSWORD_ADMIN, AUDIT_ADMIN, ...

Рис 2.5.1 — створення нових користувачів

2.6 Створення збережених процедур

Збережена процедура – це набір команд T-SQL, яка зберігається на сервері і представляє собою самостійний об'єкт. Збережена процедура існує незалежно від таблиць або інших об'єктів баз даних.

Розроблено процедуру, що повертає всіх працівників Hilton Hotel (див. рис. 2.6.1). Результати відпрацювання процедури на рисунку 2.6.2.

```
38 CREATE PROCEDURE showHiltonEmployee()
39 BEGIN
40     select name, surname, positionID
41     from employee
42     where positionID = (select id from hotel where hotel.name = 'Hilton');
43 END;
44
45
46 ✓ call showHiltonEmployee;
```

Рис. 2.6.1 — Означення процедури та її негайний виклик

	name	surname	positionID
1	alexuk	yana	1
2	habib	nasran	1

Рис. 2.6.2 — Відпрацювання процедури *showHiltonEmployee()*

На рисунку 2.6.3 показано створення процедури виселення мешканця:

```

52 create procedure evictInhabitant(currentClientID int)
53 begin
54     update orders
55     set livingDays = 0
56     where orders.clientID = currentClientID;
57 end;
58
59
60 ✓ call evictInhabitant( currentClientID: 10)

```

Рис. 2.6.3 — Означення процедури *evictInhabitant()* та її негайний виклик

2.7 Створення тригерів

Тригер - це вид збереженої процедури, яку сервер бази даних автоматично викликає при виконанні операцій модифікації таблиць. Тригери використовуються для перевірки цілісності даних, а також для видачі попереджень.

Кожен тригер прив'язується до конкретної таблиці. Але він містить три недоліка: складність, вплив на продуктивність і прихована функціональність.

Створено тригер для логування замовлень. Для цього створено тестову таблицю *logs*. Коли спрацьовує процедура *evictInhabitant()*, вона модифікує таблицю *orders*, що виконує умову створеного тригеру [12].

```

63 -- logs
64 CREATE TABLE `log`
65 (
66     id      INT(11) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
67     msg     VARCHAR(255)      NOT NULL,
68     time    TIMESTAMP         NOT NULL DEFAULT CURRENT_TIMESTAMP,
69     row_id  INT(11)           NOT NULL
70 );
71
72 DELIMITER |
73 CREATE TRIGGER logOrders
74     AFTER UPDATE
75     ON orders
76     FOR EACH ROW
77 BEGIN
78     INSERT INTO log Set msg = 'insert', row_id = NEW.id;
79 END;

```

Рис. 2.7.1 — Код триггеру логування logOrders

Після виклику процедури `evictInhabitant()`, оновилася таблиця `log` (див. рис. 2.7.2)

WHERE		ORDER BY		
	id	msg	time	row_id
1	1	insert	2021-12-20 19:25:22	10

Рис. 2.7.2 — Детект тригера logOrders

2.8 Створення представлень

Представлення є тимчасовими наборами даних і існують тільки тоді, коли працюють з ними. Представлення - це не копія даних, а посиланням на реальні записи. Вони дозволяють отримувати дані, які задовольняють потреби користувачів. Після створення представлень з ними можна поводитися так само, які із звичайними таблицями.

Розроблено представлення `names_view_who_has_order`, що містить імена користувачів серед усіх, що мають хоча би одне замовлення (див. рис. 2.8.1).

```
81
82 create view names_view_who_has_order as
83 select id, name, surname
84 from client
85 where (select clientID from orders where clientID = client.id) = id;
86
87 ✓ select * from names_view_who_has_order;
```

Output hotelComplex.names_view_who_has_order

	id	name	surname
1	1	ZZBYwPGbAF	nIFGdQcjdb
2	2	yACKfjvDqm	qpDIwyDKus
3	3	Sa0gDzMUxh	APUFRxLRBd
4	4	GJRhjhPhVx	RggZrYfakd

Рис. 2.8.1 — Виконання представлення names_view_who_has_order

Нижче наведено ще одне представлення, що наглядно демонструє асортимент зручностей усіх кімнат на даний момент (див рис. 2.8.2).

```
92
93 create view roomFacilities as
94 select *
95 from comfort;
96
97 ✓ select * from roomFacilities;
```

Output hotelComplex.roomFacilities

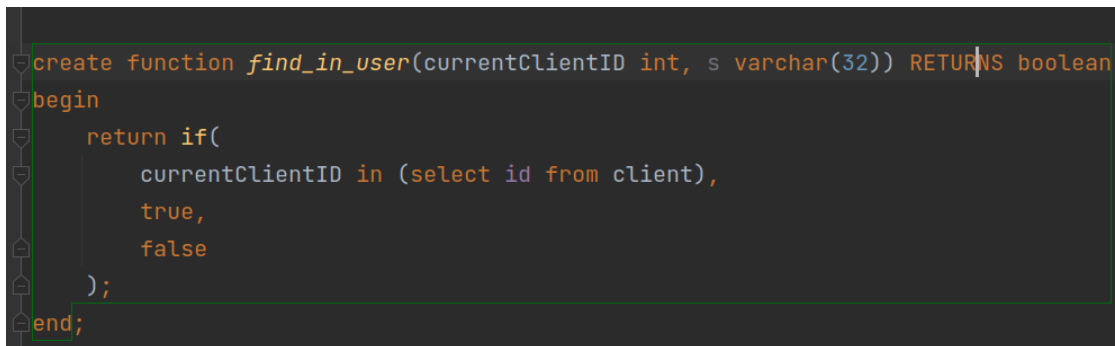
id	description
1	lux
2	norm
3	3/10
4	classic
5	half-lux

Рис. 2.8.2 — Виконання представлення roomFacilities

2.9 Створення функцій

SQL-функції виконують довільний список операторів SQL та повертають результат останнього запиту у списку. У простому випадку (не з безліччю) буде повернено перший рядок результату останнього запиту. Збережені процедури складаються з декількох інструкцій і мають від нуля до декількох вхідних параметрів, але зазвичай не повертають жодних параметрів. На відміну від процедур, що зберігаються, функції завжди повертають одне значення.

Розроблено функцію пошуку в користувачеві на рисунку 2.9.1:



```
create function find_in_user(currentClientID int, s varchar(32)) RETURNS boolean
begin
    return if(
        currentClientID in (select id from client),
        true,
        false
    );
end;
```

Рис. 2.9.1 — Функція пошуку користувача

2.10 Створення індексів

Індекс - об'єкт бази даних, створюваний з метою підвищення продуктивності пошуку даних. Таблиці в базі даних можуть мати велику кількість рядків, які зберігаються в довільному порядку, та їх пошук за заданим критерієм шляхом послідовного перегляду таблиці рядок за рядком може тривати багато часу. Індекс формується із значень одного або кількох стовпців таблиці та покажчиків на відповідні рядки таблиці і, таким чином, дозволяє шукати рядки, які відповідають критерію пошуку.

Створено індекс на поле payment таблиці orders (див. рис. 2.10.1). Це спричиняє швидший пошук по таблиці, адже тоді дані зберігаються у відсортованому вигляді по полю payment [13].

```
102      create index connector on orders(payment);
103
104
105  ✓  select payment from orders
106     where payment > 200;
107
108
```

Рис. 2.10.1 — Индекс на поле payment

РОЗДІЛ 3. DML-ЗАПИТИ

3.1 Організація вибірки інформації з бази даних

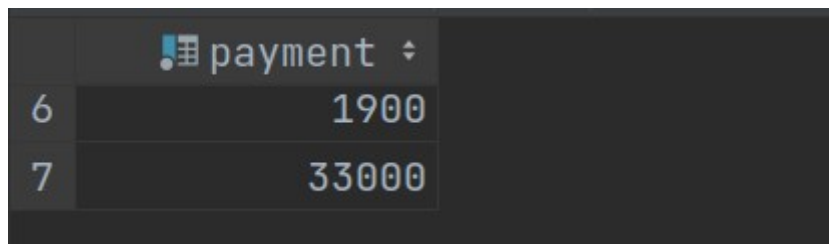
Користувача при роботі з базою даних цікавить не весь її вміст, а деяка конкретна інформація. Знайти потрібні відомості можна послідовним переглядом записів. Однак такий спосіб пошуку незручний і малоефективний особливо при великій кількості записів. Більшість систем управління базами даних дозволяють виконувати вибірку потрібної інформації шляхом виконання запитів.

Користувач відповідно до певних правил формулює запит, указуючи, якими критеріями повинна задовольняти його цікавить інформація, а система виводить записи, що задовольняють запиту. Використання запитів SQL є одним з найбільш ефективних і універсальних способів вибірки даних з таблиць бази даних.

3.2 Проста вибірка даних

Запит, який повертає всі платежі понад 200 із замовлень наведено на рисунку 3.2.1.

```
select payment  
from orders  
where payment > 200;
```

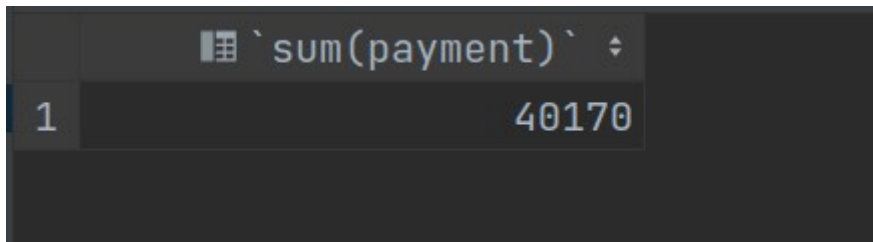


	payment
6	1900
7	33000

Рис. 3.2.1 - платежі понад 200

Запит, який повертає суму платежів всіх замовлень наведено на рисунку 3.2.2.

```
select sum(payment)
from orders;
```

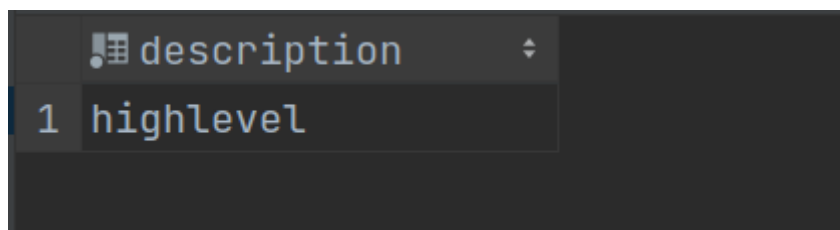


	sum(payment)
1	40170

Рис. 3.2.2 - сума платежів усіх замовлень

Запит, який повертає всі описи комфортів з довжиною понад 8 наведено на рисунку 3.2.3.

```
select description
from comfort
where length(description) > 8;
```

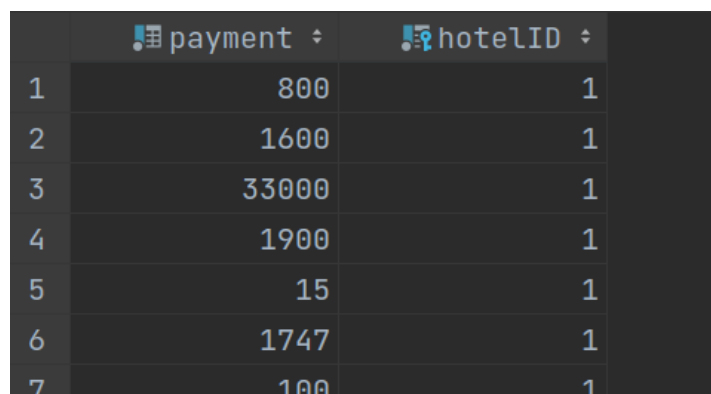


	description
1	highlevel

Рис. 3.2.3 - всі описи комфортів з довжиною понад 8

Запит, який повертає оплату в усіх готелях наведено на рисунку 3.2.4.

```
select payment, hotelID
from orders;
```



	payment	hotelID
1	800	1
2	1600	1
3	33000	1
4	1900	1
5	15	1
6	1747	1
7	100	1

Рис. 3.2.4 - оплата в усіх готелях

Запит, який повертає усіх прибиральників наведено на рисунку 3.2.5.

```
select *  
from employee  
where positionID = 6;
```

	id	name	surname	addressID	positionID	hotelID
1	2	olena	golovach	2	6	1

Рис. 3.2.5 - вибір всіх прибиральників

Запит, який повертає усі описи з довжиною менше 10 наведено на рисунку 3.2.6.

```
select description  
from position  
where description < 10;
```

Output	
# 5 довжина менше 10	
9 rows	
	description
1	shaurma seller
2	florist
3	reception
4	concierge
5	barman
6	cleaner

Рис. 3.2.6 - довжина опису менше 10

3.3 Вибірка з діапазону та множини

Запит, який повертає місто з населенням в даному діапазоні наведено на рисунку 3.3.1.

```
select name, population
from city
where population between 100 and 50000;
```

	name	population
1	Ternopil	50000
2	Kyiv	1000000

Рис. 3.3.1 - місто з населенням в наведеному діапазоні

Запит, який повертає місто з населенням в даній множині наведено на рисунку 3.3.2.

```
select name, population
from city
where population in (1000000, 50000);
```

	name	population
1	Ternopil	50000
2	Kyiv	1000000

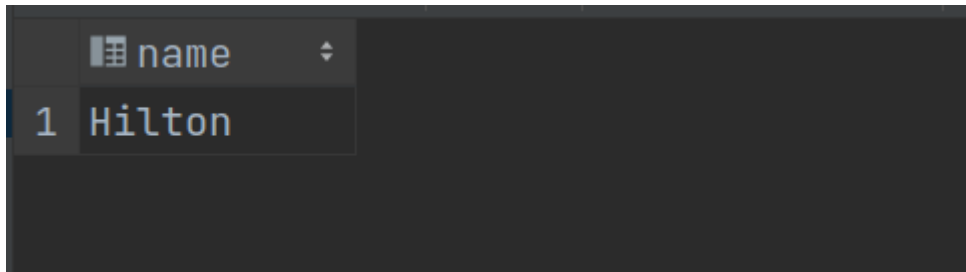
Рис. 3.3.2 - місто з населенням в наведеному діапазоні

3.4 Вибірка з регулярними виразами

Запит, який повертає готелі, що починаються на 'h' наведено на рисунку

3.4.1.

```
select name
from hotel
where name regexp '^h';
```

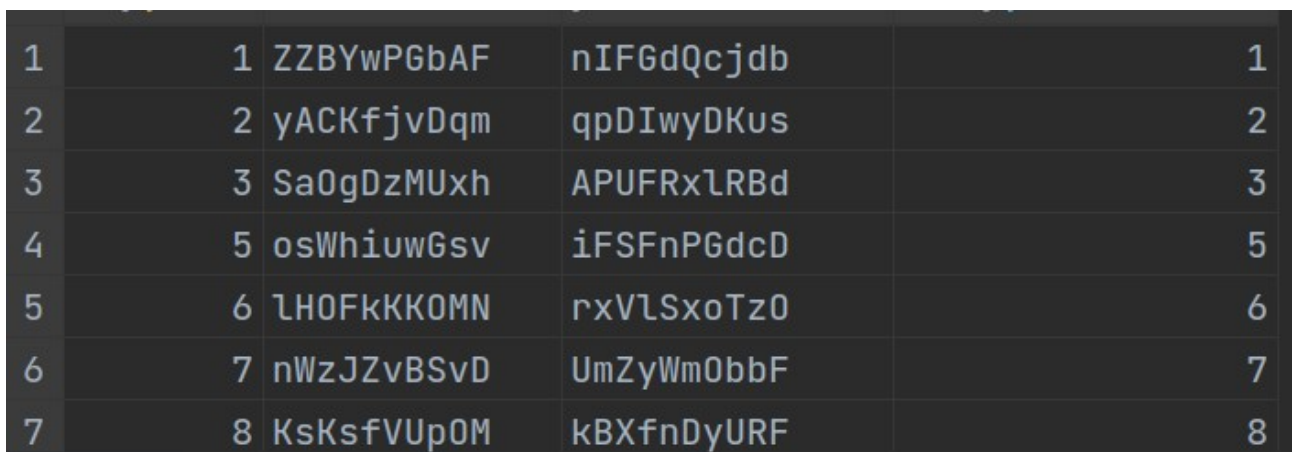


	name
1	Hilton

Рис. 3.4.1 - готелі, що починаються на 'h'

Запит, який повертає всіх невідповідних людей даному шаблону наведено на рисунку 3.4.2.

```
select *
from client
where name not regexp ('GJRhjhPhVx');
```



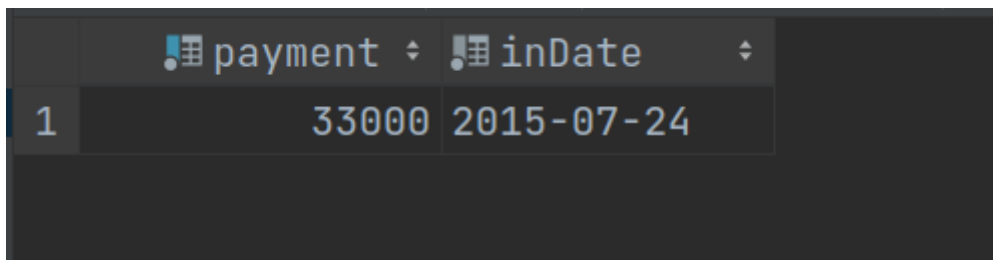
1	1	ZZBYwPGbAF	nIFGdQcjdb	1
2	2	yACKfjvDqm	qpDIwyDKus	2
3	3	Sa0gDzMUxh	APUFRx1RBd	3
4	5	osWhiuwGsv	iFSFnPGdcD	5
5	6	1H0FkKKOMN	rxVLSxoTz0	6
6	7	nWzJZvBSvD	UmZyWm0bbF	7
7	8	KsKsfVUpOM	kBXfnDyURF	8

Рис. 3.4.2 - невідповідні люди даному шаблону

3.5 Вибірка із запитамі зменшення піль

Запит, який повертає поля замовлення для найдорожчого платежу наведено на рисунку 3.5.1.

```
select payment, inDate  
from orders  
left join client c on c.id = orders.clientID  
where payment = (select max(payment) from orders);
```

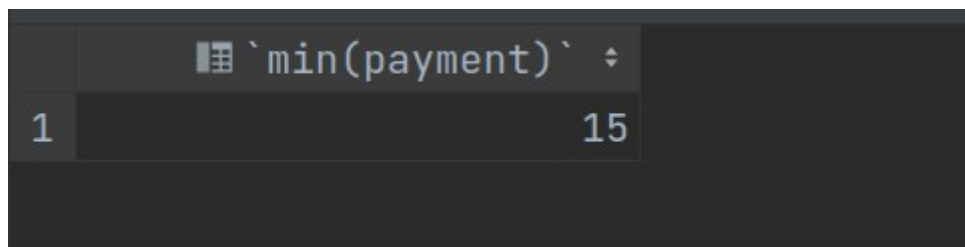


	payment	inDate
1	33000	2015-07-24

Рис. 3.5.1 - поля замовлення для найдорожчого платежу

Запит, який повертає мінімальну ціну платежу наведено на рисунку 3.5.2.

```
select min(payment)  
from orders;
```



	`min(payment)`
1	15

Рис. 3.5.2 - мінімальна ціна платежу

Запит, який повертає середню кількість населення країн наведено на рисунку 3.5.3.

```
select avg(population)  
from country;
```


	`avg(population)`
1	42000000.0000

Рис. 3.5.3 - мінімальна ціна платежу

Запит, який повертає ім'я великими літерами та прізвище малими наведено на рисунку 3.5.4.

```
select UPPER(name), LOWER(surname)
from client;
```

	`UPPER(name)`	`LOWER(surname)`
1	ZZBYWPGBAF	nifgdqcjdb
2	YACKFJVDQM	qpdiwydkus
3	SAOGDZMUXH	apufrxlrbd
4	GJRHJHPHVX	rggzryfakd
5	OSWHIUWGSV	ifsfnpgcdcd
6	LHOFKKKOMN	rxvlsxotzo

Рис. 3.5.4 - ім'я великими літерами та прізвище малими

3.6 Вибірка із групуванням

Запит, який повертає згруповані за ціною платежі наведено на рисунку 3.6.1.

```
select payment, count(*) as count
from orders
group by payment;
```

	payment	count
1	15	1
2	100	2
3	300	1
4	608	1
5	800	1
6	1600	1

Рис. 3.6.1 - згруповані за ціною платежі

Запит, який повертає працівників, згрупованих за полем айді готелю наведено на рисунку 3.6.2.

```
select hotelID, count(*) as count
from employee
group by hotelID;
```

	hotelID	count
1	1	6

Рис. 3.6.2 — згруповані за полем айді готелю працівники

3.7 Вибірка із впорядкуванням

Запит, який повертає замовлення, впорядковані за датою заселення наведено на рисунку 3.7.1.

```
select *
from orders
order by inDate;
```

id	clientID	payment	hotelID	roomID	inDate	livingDays
5	5	15	1	5	2011-10-12	15
1	1	800	1	1	2012-09-20	8
6	6	1747	1	6	2012-10-03	1
7	7	100	1	7	2013-01-24	0
2	2	1600	1	2	2013-02-06	15
8	8	608	1	8	2013-12-10	9
3	3	33000	1	3	2015-07-24	30

Рис. 3.7.1 — замовлення, впорядковані за датою заселення

Запит, який повертає вулиці, впорядковані спадним чином наведено на рисунку 3.7.2.

```
select id, description
from address
order by description desc;
```

id	description
2	Stepand Banderu 301
7	Poplavskogo 24
1	Lesi Ukrainku 30
3	Josepa Slipogo 1
5	Ivana Poluia 90
9	Ivana Franka 327
4	Gorduza Olexindra 69

Рис. 3.7.2 — вулиці, впорядковані спадним чином

3.8 Вибірка із комбінацією

Запит, який повертає комбіноване подання таблиць наведено на рисунку 3.8.1.

```
select *
from position,comfort;
```

position.id	position.description	comfort.id	comfort.description
1	shaurma seller	8	highlevel
1	shaurma seller	7	triple
1	shaurma seller	6	double
1	shaurma seller	5	half-lux
1	shaurma seller	4	classic
1	shaurma seller	3	3/10
1	shaurma seller	2	norm

Рис. 3.8.1 — комбіноване подання таблиць position та comfort

Запит, який повертає комбіноване подання таблиць наведено на рисунку

3.8.2.

```
select *  
from client, orders;
```

	client.id	name	surname	addressID	orders.id	clientID	payment	hotelID	roomID	id
1	10	iHguohEiaH	sB6KHbbBxJ	7	1	1	800	1	1	2017
2	9	EFBNzUwLYr	bTQDtWpdDf	1	1	1	800	1	1	2017
3	8	KsKsfVUpOM	kBXfnDyURF	8	1	1	800	1	1	2017
4	7	nWzJZvBSvD	UmZyWm0bbF	7	1	1	800	1	1	2017
5	6	LH0FkKKOMN	rxVLSxoTz0	6	1	1	800	1	1	2017
6	5	osWhiUwGsv	iFSFnPGdcD	5	1	1	800	1	1	2017

Рис. 3.8.2 — комбіноване подання таблиць client та orders

3.9 Вибірка inner select та union

Запит, який повертає поля замовлення з ціною менше середньої наведено на рисунку 3.9.1.

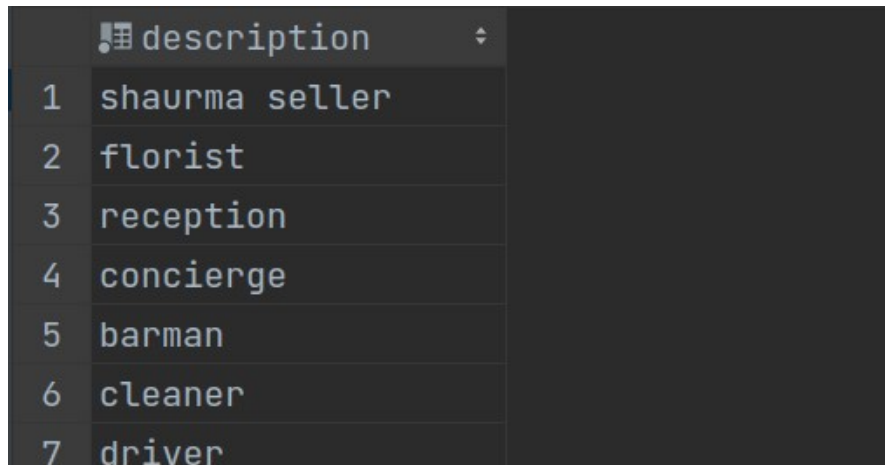
```
select clientID, payment, hotelID  
from orders  
where payment < (select AVG(payment) from orders);
```

	clientID	payment	hotelID
1	1	800	1
2	2	1600	1
3	4	1900	1
4	5	15	1
5	6	1747	1
6	7	100	1
7	8	608	1

Рис. 3.9.1 — inner select в якості правого операнда порівняння

Запит, який повертає об'єднані таблиці наведено на рисунку 3.9.2.

```
select description  
from position  
union all  
select description  
from comfort;
```



	description
1	shaurma seller
2	florist
3	reception
4	concierge
5	barman
6	cleaner
7	driver

Рис. 3.9.2 — об'єднання верхньої та нижньої таблиці

ВИСНОВКИ

Метою курсової роботи було проектування бази даних готельного комплексу. Для виконання курсової роботи були проведені всі необхідні дослідження, що стосуються розробки стратегії автоматизації та оптимізації процесу обробки даних у готелі, в результаті яких була отримана відповідь на запитання, що стосуються замовлень послуг у готельному комплексі. Спершу проведено повний аналіз предметної області, а постфактум її опис.

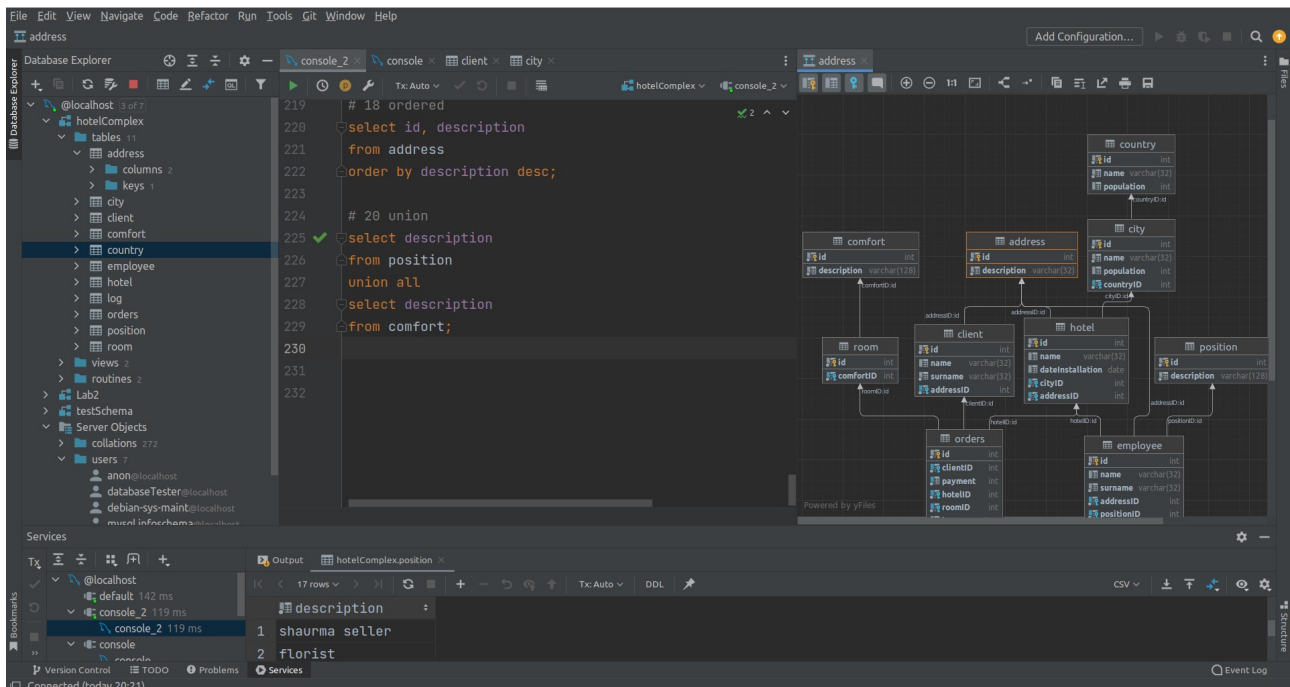
Після цього була побудована концептуальна модель для якої використано мову ER-опису предметної області, яка базується на концепції, що інформаційна модель предметної області може бути описана із застосування таких понять, як сутність, атрибут, зв'язок.

У рамках курсової роботи розроблено базу даних за заздалегідь спроектованій діаграмі. Побудовано реляційну схему бази даних на основі заданої ER-моделі, розроблено відповідні скрипти з використанням засобів мови SQL для побудови спроектованої бази даних, імпортовано дані в розроблену базу даних та виконано запити до розробленої бази даних. Цілісність даних забезпечено за допомогою обмежень та бізнес-логіки.

Розробка такої бази даних є актуальною та може бути використаною як прототип для справжнього готельного комплексу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке база даних [Електронний ресурс] – Режим доступу до ресурсу: <https://hostiq.ua/wiki/ukr/database/>.
2. Що таке СУБД – [Електронний ресурс] Режим доступу <https://brainoteka.com/courses/ms-sql-dlya-nachinayushih/chto-takoe-subd>
3. Що таке база даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.education-wiki.com/2686691-what-is-mysql-database>.
4. Що таке бази даних та чому їх вивчати? – [Електронний ресурс] / <https://saikt-online.ru/chto-takoe-bazy-dannyh-i-zachem-ix-izuchat/>
5. Проектування бази даних готельного комплексу [Електронний ресурс] – Режим доступу до ресурсу: https://knowledge.allbest.ru/programming/2c0a65625b3bd78a5c53a99421306d27_0.html.
6. Інфологічна модель [Електронний ресурс] – Режим доступу до ресурсу: https://studwood.ru/1836087/informatika/infologichna_model.
7. Інформаційно-комунікаційне забезпечення фінансової діяльності навчальний посібник/ - Х.Харківський національний економічний університет ім. С. Кузнеця, 2016. - 424 с.
8. Системи управління базами даних. Лекція – [Електронний ресурс] / Н. А. – 2020. – Режим доступу до ресурсу:
9. Пов'язування таблиць баз даних. // Інформатика / . – (https://kafinfo.org.ua/files/Informatyka_10_11/Glava_9_47.pdf). – С. 1.
10. Даталогічна модель бази даних [Електронний ресурс] – Режим доступу до ресурсу: https://studwood.ru/1553750/ekonomika/datalogichna_model_bazi_danih.
11. Створення нового користувача [Електронний ресурс] – Режим доступу до ресурсу: <https://www.digitalocean.com/community/tutorials/mysql-ru>.
12. Використання тригерів у СУБД MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://handyhost.ru/manuals/mysql/mysql-trigger.html>.
13. Індокси в MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://highload.today/indeksy-v-mysql/>.



Середовище розробки DataGrip

Апробовано іншу СУБД також. Запуск сервера PHP MY ADMIN на Linux

```
$sudo /opt/lampp/lampp start
```

✓ Показано рядки 0 - 2 (всього 3. Запит виконувався 0.0009 секунди.)

SELECT * FROM `compositions`

☐ Профілювання [[Порядкове редагування](#)] [[Редагувати](#)] [[Тлумачити SQL](#)] [[Створити PHP код](#)] [[Оновити](#)]

☐ Показати все | Число рядків: 25 | Фільтрувати рядки: Шукати в таблиці | Сортувати за ключем: Жодного

+ Параметри

				compositionName	secondDuration	viewsQuantity
<input type="checkbox"/>	Редагувати	Копіювати	Видалити	gimn vesnu	48	125845
<input type="checkbox"/>	Редагувати	Копіювати	Видалити	popliaka i znov	145	44921
<input type="checkbox"/>	Редагувати	Копіювати	Видалити	slonva kistka	194	1000845

↑ ☐ Перевірити все | Вибрані: [Редагувати](#) [Копіювати](#) [Видалити](#) [Експорт](#)

GUI СУБД PHP MY ADMIN