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) \_\_2\_\_ курсу \_ІП-13­ \_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Лисенко Анастасії Олегівни\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_Ліщук О.В. \_\_\_\_\_\_\_\_\_

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

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

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

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

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

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

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

Київ – 2022 рік

**Національний технічний університет України**

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

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-13\_\_ Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_Лисенко Анастасії Олегівні\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи \_\_\_\_\_База даних готельного комплексу\_\_\_\_\_\_\_\_\_\_\_

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

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

керівник роботи \_\_Ліщук О.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

2. Строк подання студентом роботи \_\_\_\_\_08.01.2023\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

створена база даних відповідно до умови, SQL скрипти \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

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

5) Створення користувачів бази даних

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

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

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

ER-діаграма, реляційна схема бази даних, приклади використання SQL скриптів \_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_08.11.2022**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 12.11.2022 |  |
| 2 | Побудова ER-моделі | 23.11.2022 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 5.12.2022 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 5.12.2022 |  |
| 5 | Створення користувачів бази даних | 7.12.2022 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 15.12.2022 |  |
| 7 | Створення мовою SQL запитів | 25.12.2022 |  |
| 8 | Оптимізація роботи запитів | 02.01.2023 |  |
| 9 | Оформлення пояснювальної записки | 10.01.2022 |  |
| 10 | Захист курсової роботи | 10.01.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Лисенко А.О.\_\_\_\_\_\_\_

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_**Ліщук О.В.\_\_\_\_\_\_\_

(підпис ) (прізвище та ініціали)

*Contents*

[Вступ 4](#_Toc124147211)

[1. Опис предметного середовища 5](#_Toc124147212)

[2. Постановка задачі 6](#_Toc124147213)

[3. ER-Діаграма 7](#_Toc124147214)

[3.1. Бізнес-правила: 7](#_Toc124147215)

[3.2. Вибір сутностей: 7](#_Toc124147216)

[3.3. Набори атрибутів сутностей 8](#_Toc124147217)

[4. Реляційна модель бази даних 12](#_Toc124147218)

[5. Реалізація бази даних 13](#_Toc124147219)

[6. Створення користувачів 18](#_Toc124147220)

[7. SQL запити 21](#_Toc124147221)

[7.1. Створення тригерів на таблиці, що будуть змінюватись користувачами. 21](#_Toc124147222)

[7.2. DML-запити 28](#_Toc124147223)

[7.3. Процедури та функції 42](#_Toc124147224)

[7.4. Приклад роботи індексів 49](#_Toc124147225)

[Висновок 51](#_Toc124147226)

[Посилання на джерела 52](#_Toc124147227)

# Вступ

Бази даних використовуються для зберігання та керування великими обсягами структурованих даних. Вони є важливим інструментом для багатьох компаній, оскільки дозволяють ефективно організовувати та маніпулювати даними, а також дозволяють легко отримувати певну інформацію.

Використання бази даних може значно підвищити ефективність і продуктивність бізнесу. Наприклад, компанія може використовувати базу даних для зберігання інформації про клієнтів, такої як контактні дані та історія покупок. Потім можна легко отримати доступ до цієї інформації та отримати запит для створення цільових маркетингових кампаній або надання персоналізованого обслуговування клієнтів. Крім того, централізоване зберігання всієї цієї інформації спрощує співпрацю та обмін даними всередині компанії.

Бази даних також можуть допомогти підприємствам приймати кращі рішення, надаючи доступ до точних і актуальних даних. Наприклад, компанія може використовувати базу даних для відстеження тенденцій продажів і аналізу поведінки споживачів, що може стати основою для стратегічного планування та розподілу ресурсів.

Підсумовуючи, бази даних є цінним інструментом для бізнесу, оскільки вони дозволяють організовувати, маніпулювати та отримувати великі обсяги даних, підвищуючи ефективність і обґрунтовуючи прийняття рішень. В даній курсовій роботі я виконала завдання створення бази даних готельного комплексу. Створена база даних суттєво полегшить процес роботи готельного комплексу як структури.

# Опис предметного середовища

Готельний комплекс – велика багатофункціональна будівля. Метою готельного комплексу є забезпечення проживання та обслуговування мандрівників і туристів. У базі даних готельного комплексу, як правило, є таблиці для зберігання інформації про номери, гостей, працівників, бронювання та різні інші аспекти діяльності готелю. Процедури та функції можна використовувати для виконання певних завдань, таких як створення й керування бронюваннями, обчислення рівня зайнятості номерів і створення звітів. Тригери можна використовувати для забезпечення виконання бізнес-правил і забезпечення цілісності даних. Використовуючи базу даних для зберігання та керування даними про готельний комплекс, можна легше відстежувати та аналізувати інформацію про гостей, номери та інші аспекти роботи готелю, а також приймати більш обґрунтовані рішення про те, як керувати та покращувати бізнес.

# Постановка задачі

Метою даної роботи є розробка бази даних для готельного комплексу. Тобто організація даних таким чином, щоб робота готельного комплексу виконувалась максимально ефективно. Отже основні задачі:

Гость готелю повинен мати змогу:

* Зробити бронювання на номер
* Отримати знижку за пільговою групою, якщо він до неї належить
* Заздалегідь дізнатися ціну на певний тип номеру в залежності від часу перебування в готелі
* Подивитися всі свої бронювання

Працівник готелю повинен мати змогу:

* Переглянути список вільних кімнат на певний термін часу
* Зробити бронювання для певного гостя
* Переглянути усі бронювання в готелі
* Змінити ціну кімнати за потребою

# ER-Діаграма

Після аналізу було виділено такі сутності та зв’язки між ними:

# Бізнес-правила:

* Дата в’їзду до готелю не може бути меншою за дату виїзду.
* Кожен гость та працівник готелю має бути повнолітнім.
* В одну й ту саму кімніту в один і той самий час не можуть заїхати різні клієнти.
* В готелі не може бути двох клієнтів з однаковими паспортними даними.

# Вибір сутностей:

* Гость готелю
* Робітник готелю
* Посада робітника
* Готель
* Тип кімнати
* Кімната
* Бронювання
* В’їзд
* Виїзд
* Типи знижок
* Знижки

# Набори атрибутів сутностей

Таблиця 3.1. – Набір сутностей та їх атрибутів

|  |  |
| --- | --- |
| Сутність | Атрибути |
| Booking | booking\_id  room\_id  customer\_id  checkin\_date  checkout\_date  duration |
| Booking\_discount | booking\_discount\_id  booking\_id  discount\_id |
| Check\_in | check\_in\_id  check\_in\_date  check\_out\_date  manager\_id |
| Check\_out | check\_out\_id  check\_out\_date  manager\_id |
| Customer | customer\_id  first\_name  last\_name  additional\_info  passport\_number  passport\_series  age  gender |

Продовження таблиці 3.1.

|  |  |
| --- | --- |
| Discount | discount\_id  reason\_for  percent\_amount |
| Employee | employee\_id  first\_name  last\_name  passport\_number  passport\_series  age  gender  type\_of\_employee\_id  working\_hours\_month |
| Hotel | hotel\_id  hotel\_name  capacity  location |
| Room | room\_id  room\_number  room\_type\_id  hotel\_id  housekeeper\_id |
| Room\_type | room\_type\_id  room\_name  bedroom\_amount  price\_per\_night |
| Type\_of\_employee | type\_of\_employee\_id  position\_name  wage |

Готель може мати багато номерів, тому зв’язок між таблицями «Hotel» і «Room» є «один до багатьох».

Кімната може бути пов’язана лише з одним типом кімнати, тому зв’язок між таблицями «Room» і «Room\_type» є «один до одного».

Клієнт може зробити багато бронювань, тому зв’язок між таблицями «Customer» і «Booking» є «один до багатьох».

Бронювання пов’язане з одним номером і одним клієнтом, тому зв’язок між таблицями «Booking» і «Room», а також таблицями «Booking» і «Customer» є «один до одного».

До бронювання може бути застосовано багато знижок, тому зв’язок між таблицями «Booking» та «Discount» є «один до багатьох».

Робітник готелю може бути пов’язаний із багатьма бронюваннями (або як менеджер, відповідальний за реєстрацію клієнта, або як працівник, який обробив бронювання), тому зв’язок між таблицями «Employee» і «Booking» є «один до багатьох».

Робітник готелю може бути пов’язаний лише з одним типом співробітника, тому зв’язок між таблицями «Employee» і «Type\_of\_employee» є «один до одного».

Diagram

Description automatically generated

Рисунок 3.3.1 - ER-Діаграма

# Реляційна модель бази даних

Побудова необхідних відношень та визначення первинних та зовнішніх ключів.

Diagram

Description automatically generated

Рисунок 4.1 - Реляційна модель бази даних

На даній схемі видно, що база даних знаходиться у 3 нормальній формі, адже всі поля таблиць декомпозовані, також всі атрибути таблиць функціонально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

1. Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.

# Реалізація бази даних

* 1. Створення бази даних

DROP DATABASE IF EXISTS hotel\_complex;

CREATE DATABASE hotel\_complex;

USE hotel\_complex;

DROP TABLE IF EXISTS Hotel;

DROP TABLE IF EXISTS Room\_type;

DROP TABLE IF EXISTS Room;

DROP TABLE IF EXISTS Customer;

DROP TABLE IF EXISTS Booking;

DROP TABLE IF EXISTS Discount;

DROP TABLE IF EXISTS Booking\_discount;

DROP TABLE IF EXISTS Check\_in;

DROP TABLE IF EXISTS Check\_out;

DROP TABLE IF EXISTS Employee;

DROP TABLE IF EXISTS Type\_of\_employee;

CREATE TABLE Hotel(

hotel\_id INT AUTO\_INCREMENT PRIMARY KEY,

location VARCHAR(50) NOT NULL,

hotel\_name VARCHAR(20) NOT NULL,

capacity INT NOT NULL

);

CREATE TABLE Room\_type(

room\_type\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_name VARCHAR(20) NOT NULL,

bedroom\_amount INT NOT NULL,

price\_for\_day INT NOT NULL

);

CREATE TABLE Type\_of\_employee(

type\_of\_employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

position\_name VARCHAR(20) NOT NULL,

wage INT NOT NULL

);

CREATE TABLE Employee(

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

passport\_number INT NOT NULL,

passport\_series VARCHAR(20) NOT NULL,

age INT NOT NULL,

gender VARCHAR(10) NOT NULL,

type\_of\_employee\_id INT NOT NULL,

working\_hours\_month INT NOT NULL,

FOREIGN KEY(type\_of\_employee\_id) REFERENCES Type\_of\_employee(type\_of\_employee\_id) ON DELETE CASCADE

);

CREATE TABLE Room(

room\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_type\_id INT NOT NULL,

room\_number INT NOT NULL,

hotel\_id INT NOT NULL,

housekeeper\_id INT NOT NULL,

FOREIGN KEY(housekeeper\_id) REFERENCES Employee(employee\_id) ON DELETE CASCADE,

FOREIGN KEY(hotel\_id) REFERENCES Hotel(hotel\_id) ON DELETE CASCADE,

FOREIGN KEY(room\_type\_id) REFERENCES Room\_type(room\_type\_id) ON DELETE CASCADE

);

CREATE TABLE Customer(

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

additional\_info VARCHAR(50) NOT NULL,

passport\_number INT NOT NULL,

passport\_series VARCHAR(20) NOT NULL,

age INT NOT NULL,

gender VARCHAR(11) NOT NULL

);

CREATE TABLE Check\_in(

check\_in\_id INT AUTO\_INCREMENT PRIMARY KEY,

check\_in\_date DATE NOT NULL,

check\_out\_date DATE NOT NULL,

manager\_id INT NOT NULL,

FOREIGN KEY(manager\_id) REFERENCES Employee(employee\_id) ON DELETE CASCADE

);

CREATE TABLE Check\_out(

check\_out\_id INT AUTO\_INCREMENT PRIMARY KEY,

check\_out\_date DATE ,

manager\_id INT ,

FOREIGN KEY(manager\_id) REFERENCES Employee(employee\_id) ON DELETE CASCADE

);

CREATE TABLE Booking(

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_id INT NOT NULL,

customer\_id INT NOT NULL,

checkin\_date INT NOT NULL,

checkout\_date INT NOT NULL,

duration INT,

FOREIGN KEY(room\_id) REFERENCES Room(room\_id) ON DELETE CASCADE,

FOREIGN KEY(customer\_id) REFERENCES Customer(customer\_id) ON DELETE CASCADE,

FOREIGN KEY(checkin\_date) REFERENCES Check\_in(check\_in\_id) ON DELETE CASCADE,

FOREIGN KEY(checkout\_date) REFERENCES Check\_out(check\_out\_id) ON DELETE CASCADE

);

CREATE TABLE Discount(

discount\_id INT AUTO\_INCREMENT PRIMARY KEY,

percent\_amount INT NOT NULL,

reason\_for VARCHAR(50) NOT NULL

);

CREATE TABLE Booking\_discount(

booking\_discount\_id INT AUTO\_INCREMENT PRIMARY KEY,

discount\_id INT NOT NULL,

booking\_id INT NOT NULL,

FOREIGN KEY(booking\_id) REFERENCES Booking(booking\_id) ON DELETE CASCADE,

FOREIGN KEY(discount\_id) REFERENCES Discount(discount\_id) ON DELETE CASCADE

);

ALTER TABLE Customer

ADD CONSTRAINT CheckAge CHECK(age >= 18);

ALTER TABLE Employee

ADD CONSTRAINT CheckAgeEmpl CHECK(age >= 18);

-- ALTER TABLE Customer

-- ADD CONSTRAINT CheckGenderCust CHECK(gender IN('Female', 'Male', 'Non-binary'));

ALTER TABLE Employee

ADD CONSTRAINT CheckGenderEmpl CHECK(gender IN('Female', 'Male', 'Non-binary'));

ALTER TABLE Room\_type

ADD CONSTRAINT CheckRoom\_type CHECK(room\_name IN('Normal', 'Half-Luxe', 'Luxe'));

ALTER TABLE Room\_type

ADD CONSTRAINT CheckBedroom\_amount CHECK(bedroom\_amount IN(1, 2, 3, 4));

ALTER TABLE Check\_in

ADD CONSTRAINT Date\_check CHECK(check\_in\_date < check\_out\_date);

ALTER TABLE Type\_of\_employee

ADD CONSTRAINT Wage\_check CHECK(wage >= 10);

ALTER TABLE Type\_of\_employee

ADD CONSTRAINT Type\_of\_employee\_check CHECK(position\_name IN('manager', 'housekeeper', 'porter', 'driver'));

* 1. Імпорт даних в базу даних

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/employee.csv' INTO TABLE Employee

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(first\_name,last\_name,passport\_number,passport\_series,age,gender,type\_of\_employee\_id,working\_hours\_month);

Використовуючи даний скрипт завантажую дані в базу даних з .csv файлу. Маю такі результати в таблиці:

Table

Description automatically generated

Рисунок 5.1 - приклад заповнення таблиці бази даних

Сам файл має вигляд:

Graphical user interface, application, table, Excel

Description automatically generated

Рисунок 5.2 - приклад файлу .csv

# Створення користувачів

drop user anastasiia@localhost;

drop user customer@localhost;

drop user manager@localhost;

flush privileges;

CREATE USER anastasiia@localhost IDENTIFIED BY 'developer';

CREATE USER customer@localhost IDENTIFIED BY 'customer';

CREATE USER manager@localhost IDENTIFIED BY 'manager';

GRANT ALL

ON hotel\_complex.\*

TO anastasiia@localhost;

GRANT EXECUTE

ON PROCEDURE hotel\_complex.getCustomerBookings

TO customer@localhost;

GRANT EXECUTE

ON FUNCTION hotel\_complex.calculateTotalCost

TO customer@localhost;

GRANT EXECUTE

ON PROCEDURE hotel\_complex.createCheckIn\_CheckOut

TO customer@localhost;

GRANT INSERT, UPDATE, DELETE

ON hotel\_complex.\*

TO manager@localhost;

GRANT EXECUTE

ON PROCEDURE hotel\_complex.getFreeRooms

TO manager@localhost;

GRANT EXECUTE

ON PROCEDURE hotel\_complex.createBooking

TO manager@localhost;

GRANT EXECUTE

ON PROCEDURE hotel\_complex.update\_room\_price

TO manager@localhost;

SHOW GRANTS FOR anastasiia@localhost;

SHOW GRANTS FOR customer@localhost;

SHOW GRANTS FOR manager@localhost;

Результат виконання SHOW GRANTS FOR для кожного з користувачів:

Graphical user interface, text, application

Description automatically generated

Рисунок 6.1 - привілеї адміна

Text

Description automatically generated

Рисунок 6.2 - привілеї гостя готелю

Text

Description automatically generated

Рисунок 6.3 - привілеї менеджера

# SQL запити

# Створення тригерів на таблиці, що будуть змінюватись користувачами.

DROP TRIGGER IF EXISTS check\_room\_availability;

DELIMITER $$

CREATE TRIGGER check\_room\_availability BEFORE INSERT ON Booking

FOR EACH ROW

BEGIN

DECLARE room\_id INT;

DECLARE checkin\_date DATE;

DECLARE checkout\_date DATE;

SET room\_id = NEW.room\_id;

SET checkin\_date = (SELECT ci.check\_in\_date FROM Check\_in ci WHERE ci.check\_in\_id = NEW.checkin\_date);

SET checkout\_date = (SELECT ci.check\_out\_date FROM Check\_in ci WHERE ci.check\_in\_id = NEW.checkin\_date);

-- SET checkout\_date = (SELECT ci.check\_out\_date FROM Check\_in ci WHERE ci.check\_out\_id = NEW.checkout\_date);

IF EXISTS (SELECT 1 FROM Booking b

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

WHERE b.room\_id = room\_id

AND (ci.check\_in\_date BETWEEN checkin\_date AND checkout\_date

OR ci.check\_out\_date BETWEEN checkin\_date AND checkout\_date)) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'The room is not available for the requested dates.';

END IF;

END$$

DELIMITER ;

Graphical user interface, application

Description automatically generated

Рисунок 7.1.1 - приклад роботи трігеру, який блокує додання клієнту в кімнату, яка вже зайнята

Такі значення не можна ввести, бо ми вже маємо бронювання на ці дати і відповідно кімната вже зайнята гостем готелю.

Цей тригер називається "check\_room\_availability". Він виконується перед операцією вставки в таблиці "Booking" та виконується для кожного рядка, який вставляється. Він перевіряє, чи вільний номер, указаний в операції вставки, на запитані дати на основі дат заїзду та виїзду, збережених у таблицях "Check\_in" і "Check\_out". Якщо кімната недоступна, тригер викликає помилку з відповідним повідомленням.

DROP TRIGGER IF EXISTS check\_customer\_exists;

DELIMITER $$

CREATE TRIGGER check\_customer\_exists BEFORE INSERT ON Customer

FOR EACH ROW

BEGIN

DECLARE passport\_series VARCHAR(20);

DECLARE passport\_number INT;

SET passport\_series = NEW.passport\_series;

SET passport\_number = NEW.passport\_number;

IF EXISTS (SELECT 1 FROM Customer WHERE passport\_series = passport\_series AND passport\_number = passport\_number) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'A customer with the same passport series and passport number already exists.';

END IF;

END$$

DELIMITER ;

A screenshot of a computer

Description automatically generated

Рисунок 7.1.2 - приклад роботи трігеру, який перевіряє чи вже є клієнт з серією та номером паспорту

Цей тригер викликається перед виконанням операції вставки в таблиці «Customer». Перевіряє, чи є в таблиці клієнт із такою ж серією та номером паспорта. Якщо клієнт із такою ж серією та номером паспорта вже існує, виникає помилка, і операція вставки не виконується.

DROP TABLE IF EXISTS Customer\_Modification\_Log;

CREATE TABLE Customer\_Modification\_Log (

log\_id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT NOT NULL,

modified\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

old\_first\_name VARCHAR(50) NOT NULL,

old\_last\_name VARCHAR(50) NOT NULL,

old\_additional\_info VARCHAR(50) NOT NULL,

old\_passport\_number INT NOT NULL,

old\_passport\_series VARCHAR(20) NOT NULL,

old\_age INT NOT NULL,

old\_gender VARCHAR(10) NOT NULL

);

DROP TRIGGER IF EXISTS customer\_modification\_log;

DELIMITER //

CREATE TRIGGER customer\_modification\_log

AFTER UPDATE ON Customer

FOR EACH ROW

BEGIN

INSERT INTO Customer\_Modification\_Log (customer\_id, modified\_at, old\_first\_name, old\_last\_name, old\_additional\_info, old\_passport\_number, old\_passport\_series, old\_age, old\_gender)

VALUES (OLD.customer\_id, CURRENT\_TIMESTAMP, OLD.first\_name, OLD.last\_name, OLD.additional\_info, OLD.passport\_number, OLD.passport\_series, OLD.age, OLD.gender);

END //

DELIMITER ;

Початкове значення

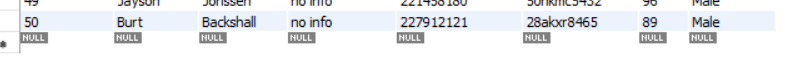


Рисунок 7.1.3 - значення до модифікації даних

Змінене значення



Рисунок 7.1.4 - після модифікації

Запис у створеній таблиці

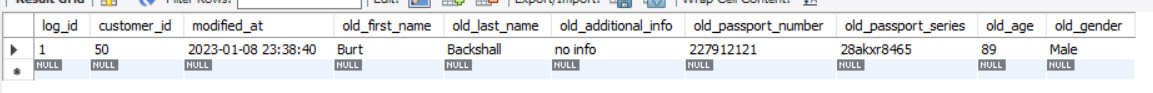


Рисунок 7.1.5 - приклад роботи трігеру, який записує усі зміни в таблиці гостя

Цей тригер називається "check\_checkout\_date" і він активується після оновлення в таблиці «Customer». Він вставляє рядок у таблицю «Customer\_Modification\_Log» із customer\_id, modified\_at, old\_first\_name, old\_last\_name, old\_additional\_info, old\_passport\_number, old\_passport\_series, old\_age і old\_gender старого запису клієнта. Це робиться за допомогою ключового слова OLD, яке посилається на старі дані в рядку, що оновлюється.

DROP TRIGGER IF EXISTS check\_checkout\_date;

DELIMITER $$

CREATE TRIGGER check\_checkout\_date BEFORE UPDATE ON Check\_out

FOR EACH ROW

BEGIN

DECLARE check\_in\_id INT;

DECLARE check\_in\_date DATE;

SET check\_in\_id = NEW.check\_out\_id;

SELECT ci.check\_in\_date INTO check\_in\_date

FROM Check\_in ci

WHERE ci.check\_in\_id = check\_in\_id;

IF NEW.check\_out\_date < check\_in\_date THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'The check-out date must be later than the check-in date.';

END IF;

END$$

DELIMITER ;

Цей тригер перевіряє той факт, що час виселення з готелю є більшим ніж час заселення.

Початкові дані:

Table

Description automatically generated

Рисунок 7.1.6 - поточні дані

При зміні дати видає помилку:

Graphical user interface, application, Teams

Description automatically generated

Рисунок 7.1.7 - приклад роботи трігеру, який блокує вставку дати виселення після дати заселення

DROP TRIGGER IF EXISTS prevent\_hotel\_deletion;

DELIMITER $$

CREATE TRIGGER prevent\_hotel\_deletion

BEFORE DELETE ON Hotel

FOR EACH ROW

BEGIN

DECLARE hotel\_id INT;

DECLARE room\_count INT;

SET hotel\_id = OLD.hotel\_id;

SELECT COUNT(\*) INTO room\_count

FROM Room

WHERE Room.hotel\_id = hotel\_id;

IF room\_count > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Deleting a hotel with rooms is not allowed.';

END IF;

END$$

DELIMITER ;

Graphical user interface, application, Teams

Description automatically generated

Рисунок 7.1.8 - приклад роботи трігеру, який блокує видалення готелю в якому є кімнати

Цей тригер призначений для запобігання видаленню готелю з таблиці Hotel, якщо в таблиці Room є номери, пов’язані з ним. Коли робиться спроба видалити рядок із таблиці Hotel, тригер виконає та підрахує кількість номерів, пов’язаних із готелем, який видаляється в таблиці Room. Якщо кількість більше 0, тригер викличе помилку, і операцію видалення не буде дозволено продовжити.

# DML-запити

SELECT c.first\_name, c.last\_name, ci.check\_in\_date, ci.check\_out\_date, b.duration, r.room\_number, rt.room\_name

FROM Customer c

INNER JOIN Booking b ON c.customer\_id = b.customer\_id

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

ORDER BY r.room\_number;

Graphical user interface, text, table

Description automatically generated

Рисунок 7.2.1 - об'єднання даних про бронювання, клієнта та його номер

Цей SELECT отримує дані з таблиць Customer, Booking, Check\_in, Room і Room\_type у базі даних.

1. SELECT c.first\_name, c.last\_name, c.passport\_number, c.passport\_series

FROM Customer c

INNER JOIN Booking b ON c.customer\_id = b.customer\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

WHERE rt.room\_name = 'Luxe' AND b.duration >= 7

ORDER BY b.duration ASC;

Graphical user interface, text, application

Description automatically generated

Рисунок 7.2.2 - дані про гостя готелю, який проживав в ньому довше 7 днів в номері типу Luxe

Цей SELECT отримує ім’я, прізвище, номер паспорта та серію паспорта клієнтів, які забронювали номер типу «Люкс» і залишилися на 7 або більше днів, і впорядковує результати за тривалістю перебування в порядку зростання.

1. SELECT b.room\_id, c.first\_name, c.last\_name, d.percent\_amount, d.reason\_for

FROM booking\_discount bd

INNER JOIN Discount d ON bd.discount\_id = d.discount\_id

INNER JOIN booking b ON bd.booking\_id = b.booking\_id

INNER JOIN customer c ON b.customer\_id = c.customer\_id

ORDER BY d.percent\_amount;  
Table

Description automatically generated

Рисунок 7.2.3 - повертає дані про гостя готелю та його знижки, якщо такі є

Цей оператор SELECT отримує дані з таблиць booking\_discount, Discount, Booking і Customer. Він повертає id номера, ім’я та прізвище клієнта, суму знижки у відсотках і причину знижки для всіх бронювань, які мають знижку. Результати впорядковані за розміром відсотка знижки.

1. SELECT COUNT(c.customer\_id) AS amount\_discounts, d.percent\_amount, d.reason\_for

FROM booking\_discount bd

INNER JOIN Discount d ON bd.discount\_id = d.discount\_id

INNER JOIN booking b ON bd.booking\_id = b.booking\_id

INNER JOIN customer c ON b.customer\_id = c.customer\_id

GROUP BY d.reason\_for;

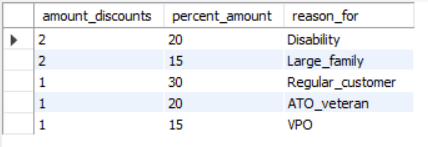


Рисунок 7.2.4 - кількість гостей з певною знижкою

Цей запит знаходить кількість клієнтів готелю з певною причиною для знижки і групує знайдені результати по причині знижки.

1. SELECT AVG(b.duration) AS average\_duration, rt.room\_name

FROM Booking b

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

GROUP BY rt.room\_name;

Graphical user interface, table

Description automatically generated

Рисунок 7.2.5 - середня довжина перебування в готелю за типами номерів

Цей оператор SELECT отримує середню тривалість усіх бронювань для кожного типу кімнати та відображає назву кімнати та середню тривалість у результатах.

1. SELECT MAX(b.duration) AS maximum\_duration, rt.room\_name

FROM Booking b

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

GROUP BY rt.room\_name

ORDER BY maximum\_duration ASC;

Graphical user interface, table

Description automatically generated with medium confidence

Рисунок 7.2.6 - максимальна довжина перебування в готелю за типами номерів

Цей запит повертає максимальну тривалість бронювання для кожного типу номеру та впорядковує набір результатів у порядку зростання максимальної тривалості.

1. SELECT SUM(b.duration) AS complete\_duration, MONTH(ci.check\_in\_date) AS Current\_month

FROM Booking b

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

GROUP BY Current\_month

ORDER BY Current\_month asc;  
Table

Description automatically generated

Рисунок 7.2.7 - загальна кількість днів, які гості готелю провели в готелі за місяцями

Цей оператор SELECT отримує загальну тривалість бронювань для кожного місяця та місяць бронювання з таблиць Booking, Check\_in, Room і Room\_type, згрупованих за поточним місяцем і впорядкованих за поточним місяцем у порядку зростання.

1. SELECT r.room\_number, rt.room\_name, rt.bedroom\_amount

FROM Room r

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

ORDER BY r.room\_number ASC;

Table

Description automatically generated

Рисунок 7.2.8 - дані про кімати

Цей оператор SELECT отримує номер кімнати, назву кімнати та кількість кімнат для кожної кімнати в таблиці Room.

1. SELECT r.room\_number, e.first\_name, e.last\_name, e.age

FROM Room r

INNER JOIN Employee e ON r.housekeeper\_id = e.employee\_id

ORDER BY e.first\_name ASC;

Table

Description automatically generated

Рисунок 7.2.9 - дані про прибиральниць для кожної кімнати

Цей оператор SQL вибере номер кімнати, ім’я, прізвище та вік кожної домробітниці в таблиці Room, відсортовані за зростанням імені господарки.

1. SELECT COUNT(e.employee\_id) AS number\_of\_employees, tof.position\_name

FROM Employee e

INNER JOIN Type\_of\_employee tof ON tof.type\_of\_employee\_id = e.type\_of\_employee\_id

GROUP BY tof.position\_name;

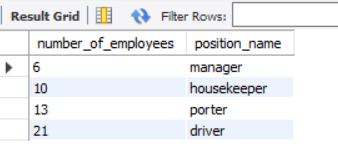


Рисунок 7.2.10 - кількість працівників за професією

Цей оператор SQL отримує кількість працівників на кожній посаді та назву посади, згруповану за назвою посади. Результати сортуються за назвою позиції.

1. SELECT SUM(e.working\_hours\_month \* tof.wage) AS total\_salary

FROM Employee e

INNER JOIN Type\_of\_employee tof ON tof.type\_of\_employee\_id = e.type\_of\_employee\_id;

Graphical user interface, application

Description automatically generated

Рисунок 7.2.11 - загальна заробітня плата

Цей оператор SELECT повертає загальну зарплату всіх працівників шляхом підсумовування добутку робочих годин кожного працівника за місяць і їхньої заробітної плати, як зазначено в таблиці Type\_of\_employee.

1. SELECT SUM(e.working\_hours\_month \* tof.wage) AS total\_salary\_for\_position, tof.position\_name

FROM Employee e

INNER JOIN Type\_of\_employee tof ON tof.type\_of\_employee\_id = e.type\_of\_employee\_id

GROUP BY tof.position\_name;

Graphical user interface, table

Description automatically generated

Рисунок 7.2.12 - заробітня плата за професією

Цей запит обчислить загальну зарплату для кожної посади шляхом підсумовування добутку робочих годин за місяць і заробітної плати для кожного працівника на цій посаді та групування результату за назвою посади.

1. SELECT SUM(ISNULL(co.check\_out\_date)) AS amount\_of\_people\_in, rt.room\_name

FROM Booking b

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

GROUP BY rt.room\_name;

Graphical user interface, application

Description automatically generated

Рисунок 7.2.13 - кількість гостей, які на даний момент знаходяться в готелі

Цей запит знаходить кількість людей, які проживають в готелі та кількість згрупована за типами кімнат. Якщо гість готелю проживає в ньому, то поле check\_out\_date в таблиці Check\_out є null-ом.

1. SELECT e.first\_name, e.last\_name, COUNT(DAYOFWEEK(ci.check\_in\_date)) AS amount\_of\_days\_checked\_in\_on\_weekends

FROM Check\_in ci

INNER JOIN Employee e ON e.employee\_id = ci.manager\_id

WHERE DAYOFWEEK(ci.check\_in\_date) in (1, 7)

GROUP BY e.last\_name;

Table

Description automatically generated

Рисунок 7.2.14 - кількість днів, кожен менеджер заселяв гостей на вихідних

Цей запит повертає список менеджерів і кількість днів, коли вони зареєстрували клієнта у вихідний день (субота чи неділя).

1. SELECT SUM(e.working\_hours\_month ) AS total\_working\_hours, tof.position\_name

FROM Employee e

INNER JOIN Type\_of\_employee tof ON tof.type\_of\_employee\_id = e.type\_of\_employee\_id

GROUP BY tof.position\_name;

Table

Description automatically generated

Рисунок 7.2.15 - загальні робочі години за професіями

Цей оператор SELECT обчислює загальну кількість робочих годин на місяць для працівників на кожній посаді.

1. SELECT b.booking\_id, c.first\_name, c.last\_name, IF(ci.check\_out\_date < co.check\_out\_date, "overstayed", "understayed") AS checked\_out\_in\_the\_specified\_time

FROM Booking b

INNER JOIN Customer c ON b.customer\_id = c.customer\_id

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

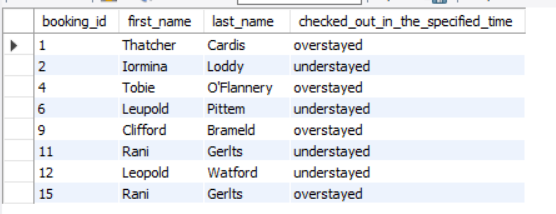
WHERE ci.check\_out\_date != co.check\_out\_date;  


Рисунок 7.2.16 - чи перевищив запланований час перебування в готелі певний клієнт

Цей запит вибирає id бронювання, ім’я клієнта, прізвище клієнта та рядок, що вказує на те, чи перевищив клієнт запланований час перебування в готелі чи навпаки.

1. SELECT COUNT(b.booking\_id) AS amount\_of\_times, IF(ci.check\_out\_date < co.check\_out\_date, "overstayed", "understayed") AS checked\_out\_in\_the\_specified\_time

FROM Booking b

INNER JOIN Customer c ON b.customer\_id = c.customer\_id

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

WHERE ci.check\_out\_date != co.check\_out\_date

GROUP BY IF(ci.check\_out\_date < co.check\_out\_date, "overstayed", "understayed");

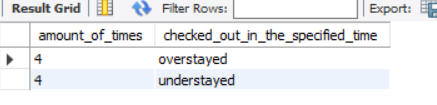


Рисунок 7.2.17 - кількість разів гість готелю затримувався в ньому або виїзджав раніше

Цей запит підрахує кількість бронювань, у яких запланована дата виїзду та фактична дата виїзду не збігаються, а потім згрупує результат за тим, чи перевищив клієнт запланований час перебування в готелі чи навпаки.

1. SELECT b.booking\_id, c.first\_name, c.last\_name, c.passport\_number,

IF(ISNULL(co.check\_out\_date) = 0, (DATEDIFF(co.check\_out\_date, ci.check\_in\_date) \* rt.price\_for\_day), (b.duration \* rt.price\_for\_day)) AS predicted\_price

FROM booking b

INNER JOIN customer c ON b.customer\_id = c.customer\_id

INNER JOIN Room r ON b.room\_id = r.room\_id

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

ORDER BY b.booking\_id ASC;

Graphical user interface, application, table

Description automatically generated

Рисунок 7.2.18 - запланова ціна за перебування

Цей оператор SELECT отримує такі дані для кожного бронювання: ідентифікатор бронювання, ім’я клієнта, прізвище клієнта, номер паспорта клієнта, прогнозована ціна бронювання.

Прогнозована ціна бронювання розраховується на основі кількості днів перебування клієнта в готелі. Якщо клієнт фізично виїхав з готелю, кількість днів розраховується як різниця між датою виїзду та датою заїзду. Якщо клієнт ще не виїхав, кількість днів береться з запланованої тривалості бронювання, яку клієнт вказує при заїзді.

1. CREATE OR REPLACE VIEW employee\_view AS

SELECT e.first\_name, e.last\_name, e.passport\_number, e.passport\_series, e.age, e.gender, e.working\_hours\_month, tof.wage, tof.position\_name, (tof.wage \* e.working\_hours\_month) AS salary

FROM Employee e

INNER JOIN Type\_of\_employee tof ON tof.type\_of\_employee\_id = e.type\_of\_employee\_id

ORDER BY salary ASC;

SELECT \* FROM employee\_view;

Graphical user interface

Description automatically generated with low confidence

Рисунок 7.2.19 - повні дані про працівника готелю

Цей запит створює представлення під назвою «employee\_view», яке відображає ім’я, прізвище, номер паспорта, серію паспорта, вік, стать, кількість робочих годин на місяць, заробітну плату, назву посади та оклад для кожного працівника в таблиці Employee.

1. CREATE OR REPLACE VIEW customer\_modification AS

SELECT CONCAT(cm.old\_first\_name, "/", c.first\_name) AS OLD\_NAME\_\_NEW\_NAME, CONCAT(cm.old\_last\_name,"/", c.last\_name) AS OLD\_SURNAME\_\_NEW\_SURNAME,

CONCAT(cm.old\_passport\_number, "/", c.passport\_number) AS OLD\_NUMBER\_\_NEW\_NUMBER, CONCAT(cm.old\_passport\_series, "/", c.passport\_series) AS OLD\_SERIES\_\_NEW\_SERIES,

CONCAT(cm.old\_age, "/", c.age) AS OLD\_AGE\_\_NEW\_AGE, CONCAT(cm.old\_gender, "/", c.gender) AS OLD\_GENDER\_\_NEW\_GENDER

FROM Customer\_modification\_log cm

INNER JOIN Customer c ON c.customer\_id = cm.customer\_id;

SELECT \* FROM customer\_modification;



Рисунок 7.2.20 - порівняння змінених даних з поточними

Цей код створить представлення під назвою customer\_modification, яке показує старі та нові значення для імені, прізвища, номера паспорта, серії паспорта, віку та статі кожного клієнта в таблиці Customer. У представленні буде показано старе значення для кожного атрибута, а потім нове значення для цього атрибута, розділене символом /.

# Процедури та функції

1. DROP FUNCTION IF EXISTS calculateTotalCost;

DELIMITER $$

CREATE FUNCTION calculateTotalCost(

roomTypeId INT,

duration INT,

discountAmount INT

) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE totalCost DECIMAL(10,2);

SELECT price\_for\_day \* duration INTO totalCost FROM Room\_type WHERE room\_type\_id = roomTypeId;

SET totalCost = totalCost - (totalCost \* (discountAmount / 100));

RETURN totalCost;

END$$

DELIMITER ;

SELECT calculateTotalCost(1, 10, 20);

Graphical user interface, text, application

Description automatically generated

Рисунок 7.3.1 - розрахунок ціни за перебування з певною знижкою

Ця функція має розрахувати загальну вартість бронювання, використовуючи такі параметри: roomTypeId: ідентифікатор типу номера, duration: тривалість перебування в днях, discountAmount: сума знижки у відсотках.



DROP PROCEDURE IF EXISTS getCustomerBookings;

DELIMITER $$

CREATE PROCEDURE getCustomerBookings(IN customerId INT)

BEGIN

SELECT \* FROM Booking WHERE customer\_id = customerId;

END$$

DELIMITER ;

CALL getCustomerBookings(12);

Graphical user interface, application

Description automatically generated

Рисунок 7.3.2 - вивід усіх бронювань певного клієнта

Ця процедура виводить усі бронювання певного гостя готелю за його індентифікатором у базі даних.

1. DROP PROCEDURE IF EXISTS getFreeRooms;

DELIMITER $$

CREATE PROCEDURE getFreeRooms(hotelId INT, startDate DATE, endDate DATE)

BEGIN

SELECT r.room\_id, r.room\_number, rt.room\_NAME

FROM Room r

INNER JOIN Room\_type rt ON r.room\_type\_id = rt.room\_type\_id

WHERE r.hotel\_id = hotelId AND r.room\_id NOT IN (

SELECT b.room\_id FROM Booking b

INNER JOIN Check\_in ci ON b.checkin\_date = ci.check\_in\_id

INNER JOIN Check\_out co ON b.checkout\_date = co.check\_out\_id

WHERE ci.check\_in\_date BETWEEN startDate AND endDate AND IF(ISNULL(co.check\_out\_date) = 0, co.check\_out\_date, ci.check\_out\_date) BETWEEN startDate AND endDate

);

END$$

DELIMITER ;

CALL getFreeRooms(1, '2023-01-07', '2023-01-21');

Table

Description automatically generated

Рисунок 7.3.3 - список вільних номерів на певну дату

Ця функція отримує список вільних номерів у вказаному готелі між датою початку та закінчення. Ідентифікатор кімнати, номер кімнати та тип кімнати повертаються в наборі результатів.



DROP PROCEDURE IF EXISTS createBooking;

DELIMITER $$

CREATE PROCEDURE createBooking(

roomId INT,

customerId INT,

checkinDate INT,

checkoutDate INT

)

BEGIN

INSERT INTO Booking (room\_id, customer\_id, checkin\_date, checkout\_date, duration) VALUES (roomId, customerId, checkinDate, checkoutDate, 0);

UPDATE booking b

INNER JOIN check\_in c on b.checkin\_date = c.check\_in\_id

SET duration = DATEDIFF(c.check\_out\_date, c.check\_in\_date) WHERE (b.booking\_id > 0);

END$$

DELIMITER ;

Graphical user interface, application

Description automatically generated

Рисунок 7.3.4 - створення бронювання

DROP PROCEDURE IF EXISTS createCheckIn\_CheckOut;

DELIMITER $$

CREATE PROCEDURE createCheckIn\_CheckOut(

checkinDate DATE,

checkoutDate DATE

)

BEGIN

INSERT INTO Check\_in (check\_in\_date, check\_out\_date, manager\_id) VALUES (checkinDate, checkoutDate, (SELECT t1.employee\_id FROM employee AS t1 JOIN (SELECT employee\_id FROM employee where type\_of\_employee\_id = 1 ORDER BY RAND() LIMIT 1) as t2 ON t1.employee\_id=t2.employee\_id));

SET @checkinId = LAST\_INSERT\_ID();

INSERT INTO Check\_out (check\_out\_date, manager\_id) VALUES (null, null);

SET @checkoutId = LAST\_INSERT\_ID();

END$$

DELIMITER ;

Table

Description automatically generated

Рисунок 7.3.5 - створення запису про заселення

Table

Description automatically generated

Рисунок 7.3.6 - створення запису про виселення

Перша процедура, createBooking, створює новий запис у таблиці Booking. Вона також розраховує тривалість бронювання та оновлює тривалість у тому самому записі. Друга процедура, createCheckIn\_CheckOut, створює новий запис у таблицях Check\_in і Check\_out. У таблиці Check\_in він також встановлює для стовпця manager\_id значення випадкового Employee\_ID із type\_of\_employee\_id, рівним 1(manager). У таблиці Check\_out для стовпців check\_out\_date та manager\_id встановлюється значення null. Він також зберігає check\_in\_id і check\_out\_id вставлених записів у змінних @checkinId і @checkoutId відповідно. Ці дві процедури працюють в тандемі. Тобто спочатку ми створюємо певні записи в таблицях Check\_in, Check\_out і далі користуємося ними для створення бронювання. Розбиття на дві процедури було створене, щоб кожен раз коли трігер на додання бронювання спрацьовував ми не додавали повторювані записи в Check\_in та Check\_out.

1. DROP PROCEDURE IF EXISTS update\_room\_price;

DELIMITER //

CREATE PROCEDURE update\_room\_price(IN p\_room\_type\_id INT, IN p\_new\_price DECIMAL(10,2))

BEGIN

DECLARE room\_type\_name VARCHAR(20);

SELECT room\_name INTO room\_type\_name

FROM Room\_type

WHERE room\_type\_id = p\_room\_type\_id;

UPDATE Room\_type

SET price\_for\_day = p\_new\_price

WHERE room\_type\_id = p\_room\_type\_id;

END;

//

DELIMITER ;

Table

Description automatically generated

Рисунок 7.3.7 - поточна ціна за номер

Graphical user interface, text, application, email

Description automatically generated

Рисунок 7.3.8 - змінена ціна

# Приклад роботи індексів

Індекси використовуються для підвищення швидкості операцій пошуку даних у базі даних. Вони роблять це, надаючи альтернативний спосіб пошуку даних у таблиці замість сканування всієї таблиці. Це особливо корисно для таблиць із великою кількістю рядків, оскільки це може значно скоротити кількість часу, необхідного для пошуку в таблиці. Індекси також можуть допомогти забезпечити унікальність даних у таблиці та використовувати їх для встановлення обмежень на дані. Загалом, індекси можуть значно підвищити продуктивність бази даних і зробити її ефективнішою.

Давайте перевіримо на практиці як працюють індекси. Виконаємо пошук запису в таблиці без та з використанням індексу.

Початковий запит:

SELECT DISTINCT customer\_id, last\_name

FROM customer

WHERE first\_name = ‘Clifford’;

Створення індексу на поле first\_name:

CREATE INDEX first\_name ON customer(first\_name);

За допомогою EXPLAIN ANALYZE бачимо таку статистику:

Перед створенням індексу:



Рисунок 7.4.1 - результат EXPLAIN на запиті до використання індексів

Після:



Рисунок 7.4.2 - результат EXPLAIN на запиті після використання індексів

Бачимо, що запит виконався в рази швидше і було переглянуто менше рядків записів.

З даних результатів очевидно, що використання індексів – це невід’ємна частина роботи з базою даних.

# Висновок

Для створення бази даних для готельного комплексу було зроблено декілька кроків. По-перше, були створені необхідні таблиці для зберігання інформації про готель, такої як номери, співробітники, клієнти та бронювання. Первинний і зовнішній ключі були визначені для забезпечення цілісності даних і встановлення зв’язків між таблицями.

Далі були створені представлення даних, щоб представити дані певним чином і полегшити запити. Також було створено помад 20 DML-запитів за допомогою яких можна дивитися деякі статистичні дані пов’язані з розробленою базою даних. Нарешті, збережені процедури та функції були реалізовані для виконання конкретних завдань, таких як обчислення загальної вартості бронювання або повернення списку вільних номерів протягом певного періоду часу. Загалом, створення бази даних готельного комплексу вимагало ретельного планування та використання команд SQL для структурування та обробки даних.

# Посилання на джерела

1. Довідник базових функцій MySQL [Електронний ресурс] – Режим доступу до ресурсу [W3Schools](https://www.w3schools.com/sql/sql_create_db.asp)
2. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу [Documentation](https://dev.mysql.com/doc/)
3. MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу [Workbench](https://www.mysql.com/products/workbench/)
4. Mockroo. Генерація даних [Електронний ресурс] – Режим доступу до ресурсу [Mockaroo](https://www.mockaroo.com/)