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 \_\_2\_\_ курсу \_ІП-11 групи

спеціальності 121 «Інженерія програмного забезпечення»

Тихонова Федора Сергійовича\_\_\_\_\_\_\_

(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»**

Інформатики та обчислювальної техніки

Факультет

Інформатики та програмної інженерії

Кафедра

Бази даних

Дисципліна

Курс 2 Група ІП-11 Семестр 1

**ЗАВДАННЯ**

**на курсову роботу студента**

**Тихонова Федора Сергійовича**

(прізвище, ім’я, по батькові)

1. Тема роботи база даних готельного комплексу

керівник роботи Ліщук О.В.

2. Строк здачі студентом закінченої роботи 16.01.2023

3. Вихідні дані до роботи створена база даних відповідно до умови, SQL скрипти

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

Опис предметного середовища, постановка задачі, побудова ER-діаграми, побудова

Реляційної моделі бази даних, реалізація бази даних, створення користувачів, SQL запити

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

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

6. Дата видачі завдання 31.10.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1. | Вступ | 31.10.2021-01.11.2021 |  |
| 2. | Опис предметного середовища | 01.11.2021-03.11.2021 |  |
| 3. | Постановка задачі | 01.11.2021-03.11.2021 |  |
| 4. | Побудова ER-моделі | 03.11.2021-13.11.2021 |  |
| 5. | Побудова реляційної схеми | 13.11.2021-17.11.2021 |  |
| 6. | Створення бази даних з допомогою обраної СУБД | 17.11.2021-19.11.2021 |  |
| 7. | Імпортування даних | 17.11.2021-19.11.2021 |  |
| 8. | Створення користувачів та реалізація їх функціоналу | 19.11.2021-24.11.2021 |  |
| 9. | Створення SQL запитів | 19.11.2021-13.12.2021 |  |
| 10. | Висновок | 13.12.2021-14.12.2021 |  |
| 11. | Перелік посилань | 31.10.2021-14.12.2021 |  |

**Студент \_\_\_\_Тихонов Ф.С.**

(підпис ) (прізвище та ініціали)

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

(підпис ) (прізвище та ініціали)

ЗМІСТ

[ВСТУП 5](#_Toc122617010)

[1 Опис предметного середовища 6](#_Toc122617011)

[2 Постановка задачі 7](#_Toc122617012)

[3 ER-діаграма 8](#_Toc122617013)

[3.1 Бізнес-правила 8](#_Toc122617014)

[3.2 Вибір сутностей 8](#_Toc122617015)

[3.3 Набори атрибутів сутностей 9](#_Toc122617016)

[4 Реляційна модель бази даних 12](#_Toc122617017)

[4.1 Побудова необхідних відношень та визначення первинних і зовнішніх ключів 12](#_Toc122617018)

[5 Реалізація бази даних 13](#_Toc122617019)

[5.1 Створення бази даних 13](#_Toc122617020)

[6 Створення користувачів бази даних 16](#_Toc122617021)

[6.1 Створення ролі менеджера бази даних: 16](#_Toc122617022)

[6.2 Створення ролі звичайного користувача бази даних: 16](#_Toc122617023)

[6.3 Створення користувача-менеджера: 16](#_Toc122617024)

[6.4 Створення звичайного користувача: 16](#_Toc122617025)

[7 SQL запити 17](#_Toc122617026)

[7.1 Створення функцій 17](#_Toc122617027)

[7.2 Створення запитів та представлень. 21](#_Toc122617028)

[7.3 Створення тригерів на таблиці clients, bookings 28](#_Toc122617029)

[7.4 Приклад роботи індексів 29](#_Toc122617030)

[Висновок 30](#_Toc122617031)

[Перелік посилань 31](#_Toc122617032)

ВСТУП

Бази даних використовуються для збереження, обробки та доступу до будь-яких даних. Вони збирають дані про людей, місця або речі. Вся ця інформація зібрана в одному місці для дослідження та спостереження. Бази даних – це свого роду організована купа даних.

Бізнеси використовують бази даних для прийняття рішень виходячи з правильної інформації. Способи використання баз даних організаціями наступні:

* Покращити бізнес-процеси. Компанії збирають дані про такі бізнес-процеси, як продаж, обробка замовлень, та клієнтські послуги.
* Мати дані клієнтів. Бази даних часто зберігають інформацію про людей, наприклад про клієнтів та користувачів. Наприклад, соціальні мережі використовують бази даних для збереження такої інформації як імена, адреси електронної пошти, та поведінка користувача.

В цій курсовій роботі я розробив базу даних для готельного комплексу. Туризм був, є і буде одним із найперспективніших бізнесів у світі. А використання баз даних може сильно полегшити роботу з клієнтами, замовленнями, тур-агентствами і так далі.

# Опис предметного середовища

Наразі найрозповсюдженим місцем для відпочинку майже у всіх країнах цивілізованого світу є готель. Їхні переваги є значними: безпечні, зручні, прості для бронювання, широкий спектр послуг, та завжди знайдеться щось для кожного: дорогі та люкс-готелі або бюджетні; All-inclusive та готелі спортивного відпочинку. Заради спрощення процесу бронювання номерів готелю варто зробити єдиний алгоритм реєстрації для клієнтів: коли він приїде, поїде, яка вартість бронювання, який спосіб оплати і так далі. Для збереження даних клієнтів добре підійде таблиця. При великих обсягах даних, буде важко спостерігати за всіма даними, що є проблемою. Але база даних дуже легко впорається з цією проблемою. Наприклад, коли декілька клієнтів бронюють один і той самий номер, треба, щоб дати бронювання не збігалися.

# Постановка задачі

Метою даної роботи є розробка бази даних для готельного комплексу. Заради швидкодії процесу реєстрації, потрібно визначити кроки алгоритму, які проводить клієнт та готельний бізнес.

Готельний бізнес:

* Зробив контракт з певними тур-агентствами, для більшого набору клієнтів, в результаті чого агентство бере комісію із замовлень. В свою чергу робітники агентств використовують свою харизму для привертання клієнтів.
* Володіє багатьма готелями, кожен готель має робітників; місця для відпочинку, наприклад спортзал, ресторан; деякий транспорт для потреб клієнтів; номери кожен з яких має декілька кімнат.

Клієнт:

* При першому бронюванні номера проходить реєстрацію, під час якої вказує своє ім’я, прізвище, номер телефону, тип оплати, серію паспорту та адресу електронної пошти.

Туристичне агентство:

* Створює замовлення, в яке входить номер деякого готелю, клієнт, туристичне агентство, ціна бронювання номеру, дата в’їзду та дата виїзду.

# ER-діаграма

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

## Бізнес-правила

1. Дата виїзду не може бути раніше, ніж дата в’їзду
2. При розробленні замовлення, декілька клієнтів не можуть забронювати один і той самий номер на один і той самий час.
3. Якщо дата вказана неправильно, то база даних видає помилку.

## Вибір сутностей

* Готельний бізнес
* Готель
* Номер готелю
* Працівник готелю
* Транспорт готелю
* Пункт відпочинку готелю
* Туристичне агентство
* Працівники тур-агентства
* Клієнт
* Бронювання

## Набори атрибутів сутностей

– Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| hotel\_business | id,  name |
| hotel | Id,  name  owned\_by  address  rating |
| hotel\_employees | id  name  surname  profession  wage |
| hotel transport | id  type  cost |
| facilities | id  name  type |
| travel agency | id  name  comission |
| agency employees | id  name  comission |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| room | id  hotel id  room amount  daily price  type |
| client | id  name  surname  phone  passport  credit card  e-mail |
| booking | id  client id  room id  agency id  price  check in date  check out date |

Сутність hotel\_business є ядром цієї діаграми. Атрибутами цієї таблиці є назва бізнесу та ідентифікатор.

Сутність hotel представляє собою готельний комплекс. Вона має відношення до hotel\_business як багато готелів до одного готельного бізнесу. Має атрибути ідентифікатор; назва; адреса; ідентифікатор бізнесу, якому належить готель; рейтинг.

Сутність hotel\_employees представляє працівників певного готелю. Має атрибути: ідентифікатор; ім’я; прізвище; професія; заробітня плата за годину.

Сутність hotel\_transport представляє собою певні одиниці транспорту, які належать певному готелю. Має атрибути: ідентифікатор; тип; вартість.

Сутність facilities представляє собою певне місце для відпочинку в готелі, наприклад: ресторан, басейн і тд. Пов’язана із сутністю готель. Має атрибути: ідентифікатор, ім’я, тип.

Сутність rooms представляє собою номер готелю. Пов’язана із сутністю готель. Має атрибути ідентифікатор, ідентифікатор готелю, якому належить кімната, ідентифікатор готелю, якому належить кімната, кількість кімнат у номері, вартість аренди на день, тип номеру.

Сутність travel agency являє собою тур-агентство. Має зв’язок з готельним бізнесом, оскільки за описом, тур-агентство на готельний бізнес підписали контракт. Має атрибути: ідентифікатор, ім’я, коміссія.

Сутність agency employees представляє робітників туристичного агентства. Мають атрибути: ідентифікатор, ім’я, прізвище, харизма, погодинна плата.

Сутність client представляє клієнта з наступними атрибутами: ідентифікатор, ім’я, прізвище, телефон, паспорт, номер платіжної картки, електронна пошта.

Сутність bookings представляє бронювання. Пов’язана з клієнтом, агентством на номером готелю. Має атрибути: ідентифікатор, ідентифікатор клієнта, ідентифікатор кімнати, ідентифікатор агентства, ціна бронювання, дата в’їзду та виїзду.

Diagram

Description automatically generated

– ER-діаграма

# Реляційна модель бази даних

## Побудова необхідних відношень та визначення первинних і зовнішніх ключів

A picture containing graphical user interface

Description automatically generated

– Реляційна схема 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. Визначення обмежень цілісності для спроектованих відношень. Обмеження цілісності: Рядок батьківської таблиці може бути видалений лише у тому випадку, якщо немає зовнішніх ключів, що посилаються на значення преференційного ключа цього рядка. (Реалізовується відсутністю параметра ON DELETE CASCADE
2. Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.

# Реалізація бази даних

## Створення бази даних

CREATE TABLE hotel\_business(  
 id SERIAL PRIMARY KEY ,  
 name VARCHAR(32) UNIQUE  
);  
  
CREATE TABLE clients(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32),  
 surname VARCHAR(32),  
 phone VARCHAR(12),  
 passport VARCHAR(9) UNIQUE,  
 credit\_card VARCHAR(16),  
 email VARCHAR(50)  
);  
  
CREATE TABLE travel\_agency(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32) UNIQUE,  
 contract\_with INTEGER,  
 comission DECIMAL,  
 FOREIGN KEY(contract\_with) REFERENCES hotel\_business ON DELETE CASCADE  
);  
  
CREATE TABLE hotel(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32),  
 belongs\_to INTEGER,  
 address VARCHAR(32),  
 rating DECIMAL,  
 FOREIGN KEY(belongs\_to) REFERENCES hotel\_business ON DELETE CASCADE  
);  
  
CREATE TABLE agency\_employee(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32),  
 surname VARCHAR(32),  
 works\_at INTEGER,  
 charisma VARCHAR(32),  
 wage INTEGER,  
 FOREIGN KEY(works\_at) REFERENCES travel\_agency ON DELETE CASCADE  
);  
  
CREATE TABLE transport(  
 id SERIAL PRIMARY KEY,  
 type VARCHAR(32),  
 cost INTEGER,  
 belongs\_to INTEGER,  
 FOREIGN KEY(belongs\_to) REFERENCES hotel ON DELETE CASCADE  
);  
  
CREATE TABLE facilities(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32),  
 part\_of INTEGER,  
 type VARCHAR(32),  
 FOREIGN KEY(part\_of) REFERENCES hotel ON DELETE CASCADE  
);  
  
CREATE TABLE rooms(  
 id SERIAL PRIMARY KEY,  
 is\_in INTEGER,  
 room\_amount INTEGER,  
 daily\_price INTEGER,  
 type VARCHAR(32),  
 FOREIGN KEY(is\_in) REFERENCES hotel ON DELETE CASCADE  
);  
  
CREATE TABLE hotel\_employees(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(32),  
 works\_at INTEGER,  
 profession VARCHAR(32),  
 wage INTEGER,  
 FOREIGN KEY(works\_at) REFERENCES hotel ON DELETE CASCADE  
);  
  
CREATE TABLE bookings(  
 id SERIAL PRIMARY KEY,  
 client\_id INTEGER,  
 room\_id INTEGER,  
 agency\_id INTEGER,  
 price INTEGER,  
 check\_in\_date DATE,  
 check\_out\_date DATE,  
 CHECK ( bookings.check\_out\_date > bookings.check\_in\_date),  
 FOREIGN KEY(client\_id) REFERENCES clients ON DELETE CASCADE,  
 FOREIGN KEY(room\_id) REFERENCES rooms ON DELETE CASCADE,  
 FOREIGN KEY (agency\_id) REFERENCES travel\_agency ON DELETE CASCADE  
);

# Створення користувачів бази даних

## Створення ролі менеджера бази даних:

CREATE ROLE db\_manager;  
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public to db\_manager;  
GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO db\_manager;  
SELECT table\_schema as schema, table\_name as table, privilege\_type as privilege  
FROM information\_schema.table\_privileges  
WHERE grantee = 'db\_manager';

## Створення ролі звичайного користувача бази даних:

CREATE ROLE db\_viewer;  
GRANT SELECT ON ALL TABLES IN SCHEMA public to db\_viewer;  
GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO db\_viewer;  
SELECT table\_schema as schema, table\_name as table, privilege\_type as privilege  
FROM information\_schema.table\_privileges  
WHERE grantee = 'db\_viewer';

## Створення користувача-менеджера:

CREATE USER ryan;  
ALTER USER ryan WITH ENCRYPTED PASSWORD '0000';  
GRANT db\_manager TO ryan;

## Створення звичайного користувача:

CREATE USER jeff;  
ALTER USER jeff WITH ENCRYPTED PASSWORD '0000';  
GRANT db\_viewer TO jeff;

# SQL запити

## Створення функцій

CREATE OR REPLACE FUNCTION *execute\_on\_update\_clients*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 BEGIN  
 IF NEW.name <> OLD.name THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS NAME CHANGED FROM % TO %'  
 , OLD.id, OLD.name, NEW.name;  
 end if;  
 IF NEW.surname <> OLD.surname THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS SURNAME CHANGED FROM % TO %'  
 , OLD.id, OLD.surname, NEW.surname;  
 end if;  
 IF NEW.phone <> OLD.phone THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS PHONE CHANGED FROM % TO %'  
 , OLD.id, OLD.phone, NEW.phone;  
 end if;  
 IF NEW.passport <> OLD.passport THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS PASSPORT CHANGED FROM % TO %'  
 , OLD.id, OLD.phone, NEW.phone;  
 end if;  
 IF NEW.credit\_card<> OLD.credit\_card THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS CREDIT CARD CHANGED FROM % TO %'  
 , OLD.id, OLD.credit\_card, NEW.credit\_card;  
 end if;  
 IF NEW.email<> OLD.email THEN  
 RAISE NOTICE  
 'THE CLIENT WITH ID % HAD HIS CREDIT CARD CHANGED FROM % TO %'  
 , OLD.id, OLD.email, NEW.email;  
 end if;  
 RETURN NEW;  
 END;  
 $$;  
  
CREATE OR REPLACE FUNCTION *execute\_on\_delete\_clients*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 BEGIN  
 RAISE NOTICE 'CLIENT WITH ID %, NAME %, SURNAME % WAS DELETED',  
 OLD.id, OLD.name, OLD.surname;  
 RETURN NEW;  
 END  
 $$;  
  
  
CREATE OR REPLACE FUNCTION *execute\_on\_insert\_clients*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 BEGIN  
 RAISE NOTICE 'A NEW CLIENT WITH ID %, NAME %, SURNAME % WAS ADDED',  
 NEW.id, NEW.name, NEW.surname;  
 RETURN NEW;  
 END;  
 $$;  
  
  
CREATE OR REPLACE FUNCTION *execute\_on\_update\_bookings*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 DECLARE b\_row bookings%rowtype;  
 BEGIN  
 IF NEW.client\_id <> OLD.client\_id THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD ITS CLIENT ID CHANGED FROM % TO %'  
 , OLD.id, OLD.client\_id, NEW.client\_id;  
 end if;  
 IF NEW.room\_id <> OLD.room\_id THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD ITS ROOM ID CHANGED FROM % TO %'  
 , OLD.id, OLD.room\_id, NEW.room\_id;  
 end if;  
 IF NEW.agency\_id <> OLD.agency\_id THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD ITS TRAVEL AGENCY ID CHANGED FROM % TO %'  
 , OLD.id, OLD.agency\_id, NEW.agency\_id;  
 end if;  
 IF NEW.price <> OLD.price THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD ITS PRICE CHANGED FROM % TO %'  
 , OLD.id, OLD.price, NEW.price;  
 end if;  
 IF NEW.check\_in\_date<> OLD.check\_in\_date THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD ITS CHECK IN DATE CHANGED FROM % TO %'  
 , OLD.id, OLD.check\_in\_date, NEW.check\_in\_date;  
 end if;  
 IF NEW.check\_out\_date<> OLD.check\_out\_date THEN  
 RAISE NOTICE  
 'THE BOOKING WITH ID % HAD HIS CREDIT CARD CHANGED FROM % TO %'  
 , OLD.id, OLD.check\_out\_date, NEW.check\_out\_date;  
 end if;  
 FOR b\_row IN SELECT \* FROM bookings LOOP  
 IF b\_row.room\_id = NEW.room\_id AND(b\_row.check\_in\_date >= NEW.check\_in\_date AND  
 b\_row.check\_in\_date <= NEW.check\_out\_date OR  
 b\_row.check\_out\_date >= NEW.check\_in\_date AND  
 b\_row.check\_out\_date <= NEW.check\_out\_date) THEN  
 RAISE EXCEPTION 'NEW BOOKING ID=% IS OVERLAID WITH ANOTHER ONE ID=%', NEW.id, b\_row.id;  
 END IF;  
 END LOOP;  
 RETURN NEW;  
 END  
 $$;  
  
CREATE OR REPLACE FUNCTION *execute\_on\_delete\_bookings*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 BEGIN  
 RAISE NOTICE 'BOOKING WITH ID % CLIENT ID % ROOM ID % WAS DELETED',  
 OLD.id, OLD.client\_id, OLD.room\_id;  
 RETURN NEW;  
 END;  
 $$;  
  
  
CREATE OR REPLACE FUNCTION *execute\_on\_insert\_bookings*()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
 $$  
 DECLARE b\_row bookings%rowtype;  
 BEGIN  
 RAISE NOTICE 'A NEW BOOKING WITH ID % CLIENT ID % ROOM ID % WAS ADDED',  
 NEW.id, NEW.client\_id, NEW.room\_id;  
 FOR b\_row IN SELECT \* FROM bookings LOOP  
 IF b\_row.room\_id = NEW.room\_id AND(b\_row.check\_in\_date >= NEW.check\_in\_date AND  
 b\_row.check\_in\_date <= NEW.check\_out\_date OR  
 b\_row.check\_out\_date >= NEW.check\_in\_date AND  
 b\_row.check\_out\_date <= NEW.check\_out\_date) THEN  
 RAISE EXCEPTION 'NEW BOOKING ID=% IS OVERLAID WITH ANOTHER ONE ID=%', NEW.id, b\_row.id;  
 END IF;  
 END LOOP;  
 RETURN NEW;  
 END;  
 $$;  
  
CREATE OR REPLACE FUNCTION *room\_usage\_history*(input\_room\_id INTEGER)  
 RETURNS TABLE(  
 used\_room\_id INTEGER,  
 client\_id INTEGER,  
 client\_name VARCHAR(32),  
 client\_surname VARCHAR(32),  
 check\_in\_date DATE,  
 check\_out\_date DATE  
 )  
LANGUAGE plpgsql  
AS $$  
BEGIN  
 RETURN QUERY  
 SELECT \* FROM rooms\_and\_clients WHERE room\_id = input\_room\_id ORDER BY check\_in\_date;  
END  
$$;  
  
CREATE OR REPLACE FUNCTION *calculate\_price\_per\_stay*(days\_num INTEGER, input\_room\_id INTEGER)  
RETURNS INTEGER  
 LANGUAGE plpgsql  
AS $$  
 DECLARE stay\_price INTEGER := 0;  
 daily\_room\_price INTEGER := 0;  
 BEGIN  
 SELECT daily\_price FROM rooms WHERE id = input\_room\_id INTO daily\_room\_price;  
 stay\_price := daily\_room\_price \* days\_num;  
 RETURN stay\_price;  
 END  
 $$;  
  
SELECT \* FROM *calculate\_price\_per\_stay*(120, 13);  
SELECT \* FROM *room\_usage\_history*(14);

## Створення запитів та представлень.

--1  
SELECT bookings.room\_id, clients.id,  
 clients.name, clients.surname,  
 bookings.check\_in\_date, bookings.check\_out\_date  
 FROM bookings  
 INNER JOIN clients on bookings.client\_id = clients.id  
 ORDER BY room\_id;

Graphical user interface, text

Description automatically generated

--2  
SELECT hotel\_business.name,  
 travel\_agency.name,  
 travel\_agency.comission  
FROM hotel\_business  
INNER JOIN travel\_agency on  
 hotel\_business.id = travel\_agency.contract\_with  
ORDER BY hotel\_business.name;

Graphical user interface, application, Teams

Description automatically generated  
--3  
SELECT facilities.name,  
 facilities.type,  
 hotel.address  
FROM facilities  
INNER JOIN hotel on hotel.id = facilities.part\_of  
WHERE facilities.type = 'gym';

Text

Description automatically generated  
--4  
SELECT hotel\_employees.name,  
 hotel\_employees.profession,  
 hotel.address  
FROM hotel\_employees  
INNER JOIN hotel on hotel.id = hotel\_employees.works\_at  
ORDER BY profession;

Graphical user interface, text, application

Description automatically generated

--5  
SELECT clients.name, clients.surname  
FROM clients  
WHERE clients.id=(SELECT hotel\_employees.id  
 FROM hotel\_employees  
 WHERE id=5);

Graphical user interface, text, application

Description automatically generated  
--6  
CREATE OR REPLACE TEMP VIEW agencies\_and\_bookings AS  
SELECT travel\_agency.id AS agency\_id,  
 bookings.id AS bookings\_id  
FROM travel\_agency  
INNER JOIN bookings ON travel\_agency.id = bookings.agency\_id;

Text

Description automatically generated with medium confidence  
--7  
CREATE OR REPLACE TEMP VIEW businesses\_and\_agencies AS  
SELECT travel\_agency.id AS agency\_id,  
 hotel\_business.name AS biz\_name  
FROM travel\_agency  
INNER JOIN hotel\_business on travel\_agency.contract\_with = hotel\_business.id;

Text

Description automatically generated with medium confidence

--8  
SELECT businesses\_and\_agencies.biz\_name,  
 agencies\_and\_bookings.bookings\_id  
FROM businesses\_and\_agencies  
INNER JOIN agencies\_and\_bookings on  
 businesses\_and\_agencies.agency\_id = agencies\_and\_bookings.agency\_id  
ORDER BY biz\_name;

Text

Description automatically generated with medium confidence  
--9  
SELECT hotel.name,  
 hotel.address,  
 rooms.id,  
 rooms.room\_amount,  
 rooms.type  
FROM hotel  
INNER JOIN rooms on hotel.id = rooms.is\_in  
ORDER BY hotel.name, rooms.id;

Graphical user interface, text

Description automatically generated

--10  
SELECT clients.name,  
 clients.surname,  
 bookings.check\_in\_date,  
 bookings.check\_out\_date  
FROM clients  
INNER JOIN bookings on clients.id = client\_id  
ORDER BY check\_in\_date;

Graphical user interface, text

Description automatically generated  
--11  
CREATE OR REPLACE TEMP VIEW agencies\_and\_their\_employees AS  
 SELECT travel\_agency.id AS agency\_id,  
 travel\_agency.name AS agency\_name,  
 agency\_employee.name AS emp\_name,  
 agency\_employee.surname AS emp\_surname  
FROM travel\_agency  
INNER JOIN agency\_employee on travel\_agency.id = agency\_employee.works\_at;

Text

Description automatically generated

--12  
SELECT businesses\_and\_agencies.biz\_name,  
 agencies\_and\_their\_employees.emp\_name AS agency\_emp\_name,  
 agencies\_and\_their\_employees.emp\_surname AS agency\_emp\_surname  
FROM businesses\_and\_agencies  
INNER JOIN agencies\_and\_their\_employees on  
 businesses\_and\_agencies.agency\_id = agencies\_and\_their\_employees.agency\_id;

Text

Description automatically generated  
--13  
CREATE OR REPLACE TEMP VIEW hotel\_and\_its\_employees AS  
 SELECT hotel\_employees.name AS emp\_name,  
 hotel\_employees.profession,  
 hotel.name AS hotel\_name,  
 hotel.address  
FROM hotel\_employees  
INNER JOIN hotel on hotel.id = hotel\_employees.works\_at  
ORDER BY profession;

Graphical user interface, text, application

Description automatically generated  
--14  
CREATE OR REPLACE TEMP VIEW business\_and\_its\_hotels AS  
 SELECT hotel\_business.name AS biz\_name,  
 hotel.name AS hotel\_name,  
 hotel.address AS hotel\_address  
 FROM hotel\_business  
INNER JOIN hotel on hotel\_business.id = hotel.belongs\_to;

Graphical user interface, application

Description automatically generated  
--15  
SELECT business\_and\_its\_hotels.biz\_name AS business,  
 hotel\_and\_its\_employees.emp\_name  
 FROM business\_and\_its\_hotels  
INNER JOIN hotel\_and\_its\_employees  
 on business\_and\_its\_hotels.hotel\_name = hotel\_and\_its\_employees.hotel\_name  
ORDER BY business, emp\_name;

Text

Description automatically generated  
--16  
CREATE OR REPLACE VIEW rooms\_and\_clients AS  
 SELECT bookings.room\_id, clients.id,  
 clients.name, clients.surname,  
 bookings.check\_in\_date, bookings.check\_out\_date  
 FROM bookings  
 INNER JOIN clients on clients.id = bookings.client\_id  
 ORDER BY room\_id, check\_in\_date;

Graphical user interface

Description automatically generated  
--17  
  
CREATE OR REPLACE TEMP VIEW hotels\_and\_facilities AS  
 SELECT hotel.name AS hotel\_name,  
 facilities.name AS fac\_name,  
 facilities.type  
 FROM hotel  
INNER JOIN facilities on hotel.id = facilities.part\_of;

Graphical user interface, text, application

Description automatically generated  
--18  
SELECT business\_and\_its\_hotels.biz\_name,  
 hotels\_and\_facilities.fac\_name AS fac\_name,  
 hotels\_and\_facilities.type AS fac\_type  
FROM business\_and\_its\_hotels  
INNER JOIN hotels\_and\_facilities on  
 business\_and\_its\_hotels.hotel\_name = hotels\_and\_facilities.hotel\_name;

Graphical user interface, application

Description automatically generated  
  
--19  
SELECT bookings.room\_id, clients.id,  
 clients.name, clients.surname,  
 bookings.check\_in\_date, bookings.check\_out\_date  
 FROM bookings  
 INNER JOIN clients on bookings.client\_id = clients.id  
 WHERE client\_id > 1778  
 ORDER BY room\_id;

Graphical user interface, text

Description automatically generated

--20  
SELECT hotel\_business.name,  
 travel\_agency.name,  
 travel\_agency.comission  
FROM hotel\_business  
INNER JOIN travel\_agency on  
 hotel\_business.id = travel\_agency.contract\_with  
WHERE comission>=3  
ORDER BY hotel\_business.name;

Graphical user interface, application

Description automatically generated

## Створення тригерів на таблиці clients, bookings

CREATE TRIGGER client\_update  
 AFTER UPDATE  
 ON clients  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_update\_clients*();  
  
CREATE TRIGGER client\_delete  
 AFTER DELETE  
 ON clients  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_delete\_clients*();  
  
CREATE TRIGGER client\_insert  
 AFTER INSERT  
 ON clients  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_insert\_clients*();  
  
CREATE TRIGGER bookings\_update  
 AFTER UPDATE  
 ON bookings  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_update\_bookings*();  
  
CREATE TRIGGER booking\_insert  
 AFTER INSERT  
 ON bookings  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_insert\_bookings*();  
  
  
CREATE TRIGGER booking\_delete  
 AFTER DELETE  
 ON bookings  
 FOR EACH ROW  
 EXECUTE PROCEDURE *execute\_on\_delete\_bookings*();

## Приклад роботи індексів

Для прикладу розглянемоо простий SQL запит типу:

SELECT \* FROM clients WHERE name = 'client\_name\_1001';

За допомогою цього запиту можна знайти інформацію про певного клієнта за його ім’ям.

Зробивши Table Scan на цій таблиці, можемо побачити дані про процес запиту.

Graphical user interface, text

Description automatically generated

Створимо індекс:

CREATE INDEX client\_name\_index ON clients(name);

Тепер проведемо Index Scan для запиту:

Text

Description automatically generated

Можемо побачити, що вартість запиту знизилася з 55.12 до 8.29, що є більше ніж у 6 разів швидше.

Висновок

За цю роботу, ми створили базу даних, що є одним із найважливіших навичок для кожного розробника. Тематика цієї бази даних – це створення бази даних для готельного комплексу.

Підсумовуючи, ми розробили:

* + Опис предметного середовища
  + ER-діаграму
  + Реляційну схему
  + Створили базу даних
  + Тригери
  + Функції
  + Запити для будь-яких потреб
  + Генератори

Перелік посилань

1. [https://www.postgresql.org/](https://www.postgresql.org/docs/current/sql-createtrigger.html)
2. <https://www.postgresqltutorial.com/>
3. <https://www.tutorialspoint.com/postgresql/>
4. <https://stackoverflow.com/questions>