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 лікувального закладу

Студента (ки) \_\_2\_\_ курсу \_ІП-13\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Бондаренка Максима Вікторовича\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник: \_Ліщук Олександр Васильович\_\_\_\_

(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**

\_\_\_\_\_\_\_Бондаренку Максиму Вікторовичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи: База даних лікувального закладу

керівник роботи: Ліщук Олександр Васильович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 25.01.2023

3. Вихідні дані до роботи завдання на розробку бази даних лікувального закладу

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 СУБД
5. Створення користувачів бази даних
6. Імпорт даних з використанням засобів СУБД в базу даних
7. Створення мовою SQL запитів
8. Оптимізація роботи запитів

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

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 | 19.01.23 |  |
| 2 | Побудова ER-моделі | 19.01.23 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 20.01.23 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 20.01.23 |  |
| 5 | Створення користувачів бази даних | 20.01.23 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 21.01.23 |  |
| 7 | Створення мовою SQL запитів | 23.01.23 |  |
| 8 | Оптимізація роботи запитів | 24.01.23 |  |
| 9 | Оформлення пояснювальної записки | 25.01.23 |  |
| 10 | Захист курсової роботи | 26.01.23 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_** Бондаренко М.В.

(підпис ) (прізвище та ініціали)

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

(підпис ) (прізвище та ініціали)

ЗМІСТ

[ВСТУП 5](#_Toc125575820)

[1 Опис предметного середовища 6](#_Toc125575821)

[2 Постановка задачі 7](#_Toc125575822)

[3 ER-діаграма 8](#_Toc125575823)

[3.1 Бізнес-правила 8](#_Toc125575824)

[3.2 Вибір сутностей 8](#_Toc125575825)

[3.3 Набори атрибутів сутностей 9](#_Toc125575826)

[4 Реляційна модель бази даних 12](#_Toc125575827)

[4.1 Побудова необхідних відношень, визначення первинних та зовнішніх ключів 12](#_Toc125575828)

[5 Реалізація бази даних 13](#_Toc125575829)

[5.1 Створення бази даних 13](#_Toc125575830)

[5.2 Імпортування даних 18](#_Toc125575831)

[6 Створення користувачів бази даних 20](#_Toc125575832)

[6.1 Адміністратор 20](#_Toc125575833)

[6.2 Лікар 20](#_Toc125575834)

[6.3 Пацієнт 20](#_Toc125575835)

[7 SQL запити 21](#_Toc125575836)

[7.1 Створення тригерів на таблиці 21](#_Toc125575837)

[7.2 Створення процедур 25](#_Toc125575838)

[7.3 Створення функцій 28](#_Toc125575839)

[7.4 Створення представлень 30](#_Toc125575840)

[7.5 Створення різних запитів 32](#_Toc125575841)

[7.6 Створення індексів 43](#_Toc125575842)

[Висновок 44](#_Toc125575843)

[Перелік посилань 45](#_Toc125575844)

ВСТУП

Бази даних стали невід’ємною частиною сучасних обчислювальних систем, надаючи життєво важливу послугу для зберігання та керування великими обсягами даних. Вони широко використовуються в багатьох галузях і додатках і відіграють вирішальну роль в організації, пошуку та аналізі структурованої інформації.

Бази даних надають організаціям інструменти, необхідні для ефективного зберігання та отримання цієї інформації, вони також пропонують потужні інструменти для обробки та аналізу даних. Крім того, вони забезпечують послідовність і точність даних. Варто відмітити, що також присутні заходи безпеки, такі як автентифікація користувачів і шифрування даних для захисту конфіденційної інформації.

У цьому курсі ми заглибимося в основи баз даних, включаючи їх дизайн, структуру та функціональність, а також дослідимо використання ефективного сховища даних при розробці бази даних лікувального закладу.

# Опис предметного середовища

Лікувальний заклад – це складне та динамічне середовище, яке потребує надійної та ефективної бази даних для керування та організації величезного обсягу інформації, яку вона генерує. Предметне середовище для бази даних лікувального закладу включає різні сутності, такі як Patient, Doctor, Appointment, Service, Room, Medication та ін. Кожна сутність має власний набір атрибутів і зв’язків з іншими сутностями, які потрібно фіксувати та керувати ними в базі даних.

Пацієнти є основною метою медичного закладу, і їхня інформація, така як особисті дані, історія хвороби, інформація про страхування та медичні записи, має точно зберігатися та бути легкодоступною. База даних також повинна відстежувати зустрічі пацієнтів з лікарями та медсестрами та лікування, яке вони отримують.

Лікарі є ще одними важливими суб’єктами медичного закладу. Їхня інформація, така як ім’я, спеціальність, досвід та контактна інформація, має зберігатися та бути легкодоступною. База даних також повинна відстежувати зустрічі лікарів з пацієнтами та послуги, які вони надають.

Ліки є життєво важливим аспектом догляду за пацієнтами, і інформація про них, як-от назва, дозування, доступність і ціна, має зберігатися та бути легкодоступною. База даних також повинна відстежувати ліки, призначені пацієнтам під час прийому, та їх використання.

Палати є критично важливим компонентом середовища медичного закладу, і інформацію про них необхідно відстежувати та керувати в базі даних. Сутність кімнати включає різні атрибути, такі як номер кімнати, тип, місткість, доступність. База даних повинна відстежувати заселення пацієнтів в кімнатах.

Загалом тематичне середовище бази даних для лікувального закладу передбачає керування та організацію великого обсягу інформації, пов’язаної з пацієнтами, лікарями, призначеннями, лікуванням, ліками та ін. База даних повинна мати можливість обробляти великий обсяг даних і забезпечувати легкий доступ до інформації для медичного персоналу.

# Постановка задачі

Метою роботи є розробка бази даних лікувального закладу, а саме організація даних таким чином, щоб робота закладу виконувалась максимально ефективно.

Основними задачами бази даних буде зберігання інформації, пов’язаної з лікувальним закладом та надання змоги користувачам ефективно її використовувати. Таким чином лікарі повинні мати змогу дивитись та змінювати інформацію про прийоми, виписувати ліки до кожного прийому, передивлятись повну інформацію про пацієнтів. Пацієнти в свою чергу мають вільно дивитись всю інформацію про ліки, які їм виписують, прийоми, щоб знати дату та іншу інформацію, інформацію про лікарів, щоб розуміти їх досвід, власні медичні дані та інформацію про палати, куди вони можуть лягти.

# ER-діаграма

## Бізнес-правила

1. За один прийом сплата може відбуватись лише з однієї картки.
2. В одному прийомі може бути лише один доктор та один пацієнт.
3. Пацієнт може одночасно бути заселеним лише в одну палату та дата заселення має бути раніше за дату виселення.
4. До одного прийому може бути виписано декілька видів ліків.
5. Один пацієнт має лише одну медичну картку.
6. Під час одного прийому може надаватись лише одна послуга.

## Вибір сутностей

* Patient
* Doctor
* Room
* Room Housing
* Medical Card
* Medication Appointment
* Medication
* Appointment
* Service
* Payment

## Набори атрибутів сутностей

– Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| Patient | id PRIMARY KEY  patient\_name  date\_of\_birth  sex  address  phone\_number |
| Doctor | id PRIMARY KEY  doctor\_name  speciality  years\_of\_experience  phone\_number |
| Room | id PRIMARY KEY  room\_number  type  capacity  availability |
| Room Housing | id PRIMARY KEY  patient\_id FOREIGN KEY  room\_id FOREIGN KEY  start\_date  end\_date |
| Medical Card | id PRIMARY KEY  patient\_id FOREIGN KEY  height  weight  blood\_type  allergies |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Сутність | Атрибути |
|  | medical\_history  availability\_of\_insurance |
| Medication Appointment | id PRIMARY KEY  appointment\_id FOREIGN KEY  medication\_id FOREIGN KEY |
| Medication | id PRIMARY KEY  medication\_name  dosage\_in\_mg  availability  price |
| Appointment | id PRIMARY KEY  service\_id FOREIGN KEY doctor\_id FOREIGN KEY  patient\_id FOREIGN KEY  payment\_id FOREIGN KEY  appointment\_date |
| Service | id PRIMARY KEY  service\_name  duration\_minutes  price |
| Payment | id PRIMARY KEY  payment\_date  amount  payment\_method |

Сутність Doctor має зв’язок 1:N із сутністю Appointment, оскільки один лікар може мати багато прийомів.

Сутність Payment має зв’язок 1:N із сутністю Appointment, оскільки одна транзакція може бути за декілька прийомів.

Сутність Service має зв’язок 1:N із сутністю Appointment, оскільки одна послуга може виконуватись на різних прийомах.

Сутність Medication має зв’язок 1:N із сутністю Medication Appointment, оскільки один вид ліків може виписуватись для різних прийомів.

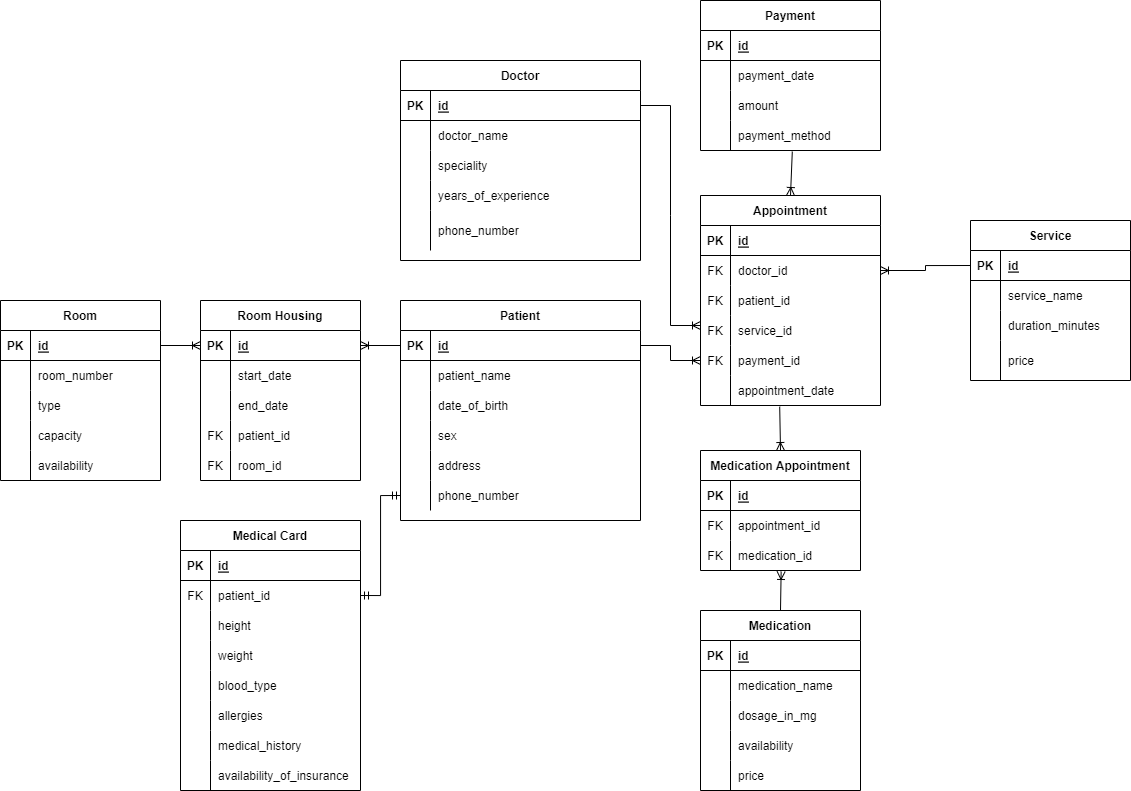
Сутність Appointment має зв’язок 1:N із сутністю Medication Appointment, оскільки до одного прийому може бути виписано декілька видів ліків.

Сутність Patient має зв’язок 1:N із сутністю Appointment, оскільки один пацієнт може бути записаний на декілька прийомів.

Сутність Patient має зв’язок 1:N із сутністю Room Housing, оскільки один пацієнт може займати різні палати в різний час.

Сутність Medical Card має зв’язок 1:1 із сутністю Patient, оскільки одна медична карта належить лише одному пацієнту.

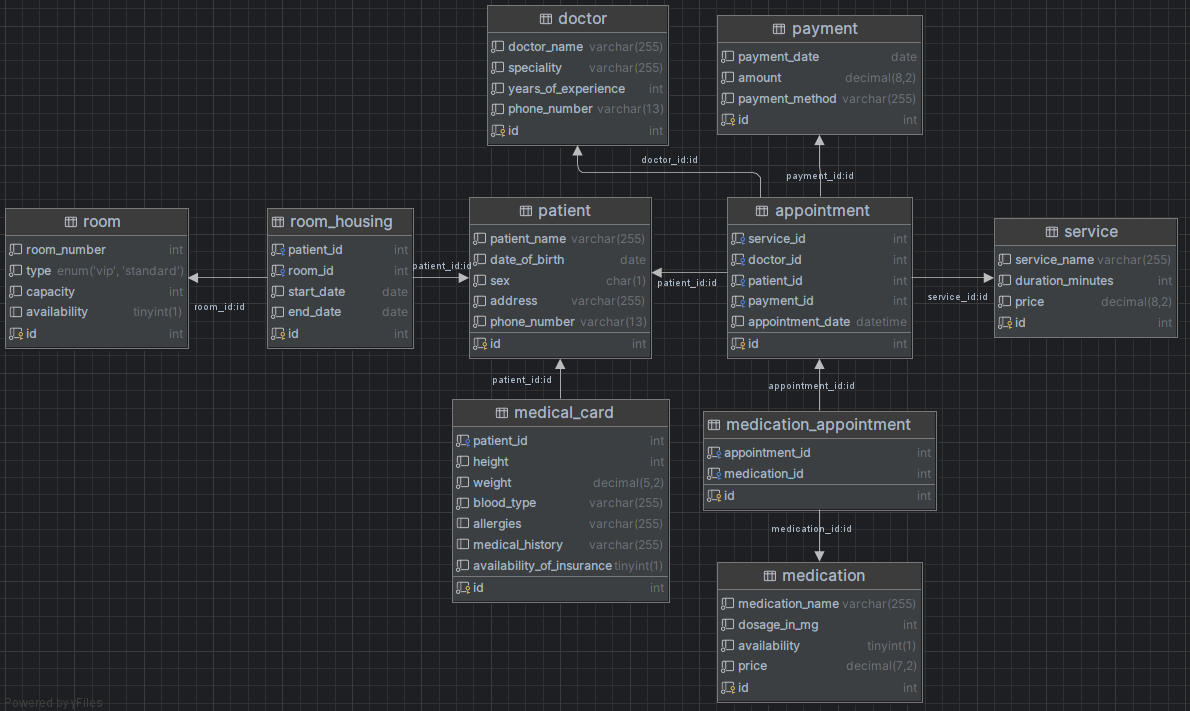
Сутність Room має зв’язок 1:N із сутністю Room Housing, оскільки одна палата може містити декілька пацієнтів.



– ER-діаграма

# Реляційна модель бази даних

## Побудова необхідних відношень, визначення первинних та зовнішніх ключів



– Реляційна схема бази даних

За схемою видно, що всі поля таблиць декомпозовані, також всі атрибути таблиць функціонально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа, отже база даних знаходиться у третій нормальній формі.

1. Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.
2. Забезпечуються каскадні дії при видаленні зовнішних ключів однієї з таблиць (ON DELETE CASCADE).

# Реалізація бази даних

## Створення бази даних

drop database hospital;  
create database if not exists hospital;  
use hospital;  
  
drop table if exists patient;  
drop table if exists doctor;  
drop table if exists room;  
drop table if exists room\_housing;  
drop table if exists medical\_card;  
drop table if exists medication\_appointment;  
drop table if exists medication;  
drop table if exists appointment;  
drop table if exists service;  
drop table if exists payment;  
  
  
create table patient  
(  
 id int auto\_increment,  
 patient\_name varchar(255) not null,  
 date\_of\_birth date not null,  
 sex char not null,  
 address varchar(255) not null,  
 phone\_number varchar(13) not null,  
 primary key (id)  
);

create table doctor  
(  
 id int auto\_increment,  
 doctor\_name varchar(255) not null,  
 speciality varchar(255) not null,  
 years\_of\_experience int not null,  
 phone\_number varchar (13) not null,  
 primary key (id)  
);  
  
create table room  
(  
 id int auto\_increment,  
 room\_number int not null,  
 type enum('VIP', 'Standard') not null,  
 capacity int not null,  
 availability bool default 1,  
 primary key (id)  
);  
  
create table room\_housing  
(  
 id int auto\_increment,  
 patient\_id int not null,  
 room\_id int not null,  
 start\_date date not null,  
 end\_date date not null,  
 primary key (id)  
);

create table medical\_card  
(  
 id int auto\_increment,  
 patient\_id int unique not null,  
 height int not null,  
 weight decimal(5,2) not null,  
 blood\_type varchar(255) not null,  
 allergies varchar(255) null,  
 medical\_history varchar(255) null,  
 availability\_of\_insurance bool not null,  
 primary key (id)  
);  
  
create table medication\_appointment  
(  
 id int auto\_increment,  
 appointment\_id int not null,  
 medication\_id int not null,  
 primary key (id)  
);  
  
create table medication  
(  
 id int auto\_increment,  
 medication\_name varchar(255) unique not null,  
 dosage\_in\_mg int not null,  
 availability bool not null,  
 price decimal(7,2) not null,  
 primary key (id)  
);  
  
create table appointment  
(  
 id int auto\_increment,  
 service\_id int not null,  
 doctor\_id int not null,  
 patient\_id int not null,  
 payment\_id int not null,  
 appointment\_date datetime not null,  
 primary key (id)  
);  
  
create table service  
(  
 id int not null,  
 service\_name varchar(255) unique not null,  
 duration\_minutes int not null,  
 price decimal(8, 2) not null,  
 primary key (id)  
);  
  
create table payment  
(  
 id int auto\_increment,  
 payment\_date date not null,  
 amount decimal(8, 2) not null,  
 payment\_method varchar(255) not null,  
 primary key (id)  
);

alter table room\_housing

add constraint room\_housing\_patient\_fk

foreign key (patient\_id) references patient(id)

on delete cascade;

alter table room\_housing

add constraint room\_housing\_room\_fk

foreign key (room\_id) references room(id)

on delete cascade;

alter table medical\_card

add constraint medical\_card\_fk

foreign key (patient\_id) references patient(id)

on delete cascade;

alter table appointment

add constraint appointment\_doctor\_fk

foreign key (doctor\_id) references doctor(id)

on delete cascade;

alter table appointment

add constraint appointment\_patient\_fk

foreign key (patient\_id) references patient(id)

on delete cascade;

alter table appointment

add constraint appointment\_service\_fk

foreign key (service\_id) references service(id)

on delete cascade;

alter table appointment

add constraint appointment\_payment\_fk

foreign key (payment\_id) references payment(id)

on delete cascade;

alter table medication\_appointment

add constraint medication\_appointment\_med\_fk

foreign key (medication\_id) references medication(id)

on delete cascade;

alter table medication\_appointment

add constraint medication\_appointment\_app\_fk

foreign key (appointment\_id) references appointment(id)

on delete cascade;

## Імпортування даних

Для заповнення даними було використано можливість імпорту з CSV файлів.

Для кожної таблиці було створено окремий файл з відповідними даними.

Перелік CSV файлів:

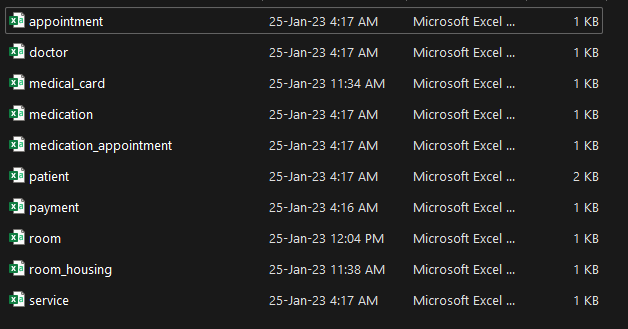


Рисунок 5.1 – CSV файли з даними

Імпорт даних здійснювався за допомогою скриптів наступного вигляду:

load data infile 'D:/KPI/DataBase/semester work/csv\_files/patient.csv' into table patient  
fields terminated by ','  
enclosed by '"'  
lines terminated by '\n'  
ignore 1 rows;

# Створення користувачів бази даних

## Адміністратор

drop role if exists administrator;  
create role administrator;  
grant all on hospital.\* to administrator;  
drop user if exists administrator@localhost;  
create user administrator@localhost identified by 'administrator';  
grant administrator to administrator@localhost;

## Лікар

drop role if exists doctor;  
create role doctor;  
grant select on hospital.\* to doctor;  
grant insert, update, delete on table hospital.medical\_card to doctor;  
grant insert, update, delete on table hospital.appointment to doctor;  
grant insert, update, delete on table hospital.medication\_appointment to doctor;  
drop user if exists doctor@localhost;  
create user doctor@localhost identified by 'doctor';  
grant doctor to doctor@localhost;

## Пацієнт

drop role if exists patient;  
create role patient;  
grant select on hospital.appointment to patient;  
grant select on hospital.medical\_card to patient;  
grant select on hospital.doctor to patient;  
grant select on hospital.room to patient;  
grant select on hospital.service to patient;  
grant select on hospital.medication to patient;  
drop user if exists patient@localhost;  
create user patient@localhost identified by 'patient';  
grant patient to patient@localhost;

# SQL запити

## Створення тригерів на таблиці

Тригер забороняє додавати пацієнтів в палату, де немає місць.

drop trigger if exists full\_rooms;

delimiter $$

create trigger full\_rooms before insert on room\_housing for each row

begin

if NEW.room\_id in (select id

from room

where availability = 0) then

signal sqlstate '45000'

set message\_text = 'The room has the maximum number of patients. Choose other room.';

end if;

end $$

delimiter ;

Приклад роботи:

Палата, що має id = 1 не має вільних місць (availability = 0):

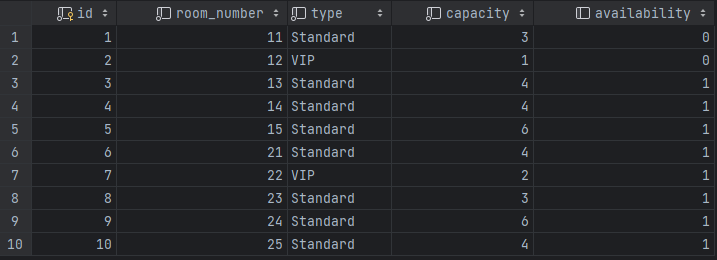


Рисунок 7.1.1 – Інформація про палати

При додаванні пацієнта в дану палату отримаємо помилку, що унеможливлює дану операцію:

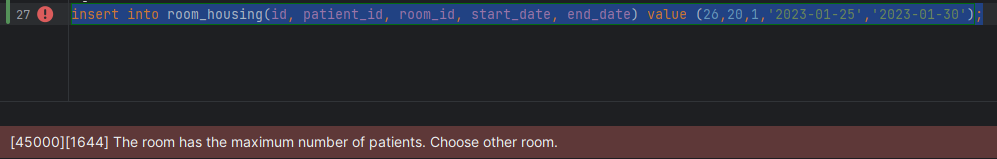


Рисунок 7.1.2 – Виконання скрипту, що активує тригер

Тригер оновлює статус палати на зайняту, коли в неї додається пацієнт, а кількість вільних місць дорівнює одному.

drop trigger if exists update\_room\_availability;

delimiter $$

create trigger update\_room\_availability before insert on room\_housing for each row

begin

if NEW.room\_id in (select distinct room\_id

from room\_housing

join room r on room\_housing.room\_id = r.id

where (select check\_free\_beds(room\_number)) = 1) then

update room

set availability = 0

where NEW.room\_id = room.id;

end if;

end $$

delimiter ;

Приклад роботи:

Зі списку палат, оберемо палату з room\_id = 3:

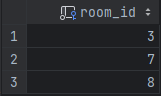


Рисунок 7.1.3 – Id палат, що мають одне вільне місце

У списку палат бачимо статус палат на даний момент (availability = 1):

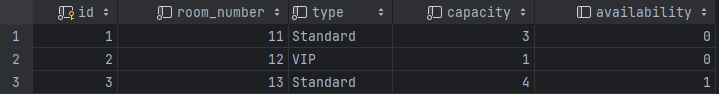


Рисунок 7.1.4 – Інформація про палати

Додаємо пацієнта в дану палату:  


Рисунок 7.1.5 – Скрипт додавання пацієнта в палату

Дивимось оновлений статус зайнятості палати:

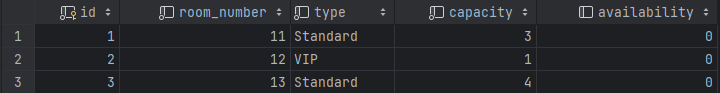


Рисунок 7.1.6 – Інформація про палати

Тригер який унеможливлює видалення старих хвороб з історії хвороб

drop trigger if exists medical\_history\_update;

delimiter $$

create trigger medical\_history\_update before update on medical\_card for each row

begin

set @history = concat(old.medical\_history, '%');

if not new.medical\_history like @history then

signal sqlstate '45000'

set message\_text = 'Old diseases must remain';

end if;

end $$

delimiter ;

Приклад роботи:

Змінимо запис в історії хвороб у медичній карті під id = 3 на ‘Asthma’:

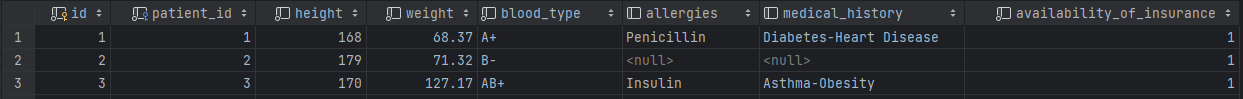


Рисунок 7.1.4 – Інформація з медичних карт пацієнтів

Отримаємо помилку, що унеможливлює зміну історії хвороб, бо хвороба, що була раніше, зникла:

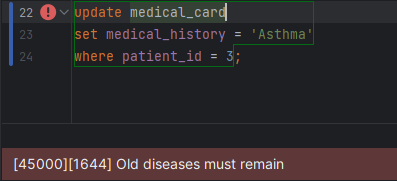


Рисунок 7.1.4 – Виконання скрипту, що активує тригер

## Створення процедур

Процедура виводить усіх пацієнтів, що живуть в палаті, id якої передається параметром.

drop procedure if exists check\_room\_patients;

delimiter $$

create procedure check\_room\_patients(IN room\_num int)

begin

select patient\_name, start\_date, end\_date

from patient

join room\_housing on patient.id = room\_housing.patient\_id

join room r on r.id = room\_housing.room\_id

where room\_number = room\_num and curdate() between start\_date and end\_date-1;

end $$

delimiter ;

call check\_room\_patients(13);

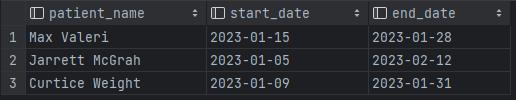


Рисунок 7.2.1 - Усі мешканці палати

Процедура показує всі прийоми, назначені на день, що передається параметром.

drop procedure if exists check\_appointments;

delimiter $$

create procedure check\_appointments(IN app\_date date)

begin

select service\_name, appointment\_date, doctor\_name, patient\_name

from appointment

join service s on appointment.service\_id = s.id

join doctor d on appointment.doctor\_id = d.id

join patient p on appointment.patient\_id = p.id

where (select date(appointment\_date))= app\_date;

end $$

delimiter ;

call check\_appointments('2023-01-20');

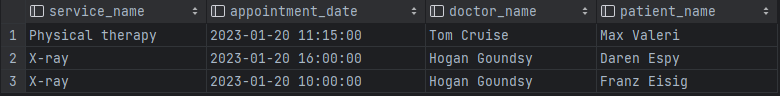


Рисунок 7.2.2 – Усі прийоми на 20.01.2023

Процедура виписує пацієнта з палати, оновлює статус палати на вільний та змінює дату кінця терміну перебування в палаті пацієнта.

drop procedure if exists discharge\_patient;

delimiter $$

create procedure discharge\_patient(IN p\_id int)

begin

update room

set availability = 1

where id = (select room\_id from room\_housing where p\_id = patient\_id and curdate() between start\_date and end\_date-1);

update room\_housing

set end\_date = curdate()

where p\_id = patient\_id and curdate() between start\_date and end\_date-1;

end $$

delimiter ;

call discharge\_patient(19);

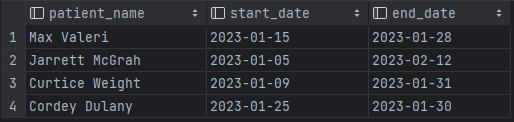


Рисунок 7.2.3 – Усі пацієнти, що записані в палаті 13 до виконання процедури

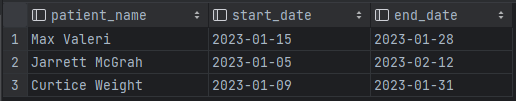


Рисунок 7.2.4 – Усі пацієнти, що записані в палаті 13 після виконання процедури



Рисунок 7.2.5 – Змінена кінцева дата перебування пацієнта, що був виписаний

## Створення функцій

Функція повертає кількість вільних місць у палаті, id якої передається параметром.

drop function if exists check\_free\_beds;  
delimiter $$  
create function check\_free\_beds(room\_num int) returns int  
 deterministic  
begin  
 declare beds\_amount int;  
 select capacity - patients\_amount into beds\_amount  
 from (select room\_number, capacity, count(\*) as patients\_amount  
 from room\_housing  
 join room r on room\_housing.room\_id = r.id  
 where curdate() between start\_date and end\_date-1  
 group by room\_id, capacity) as room\_temp  
 where room\_num = room\_number;  
 return beds\_amount;  
end $$  
delimiter ;

select check\_free\_beds(13);



Рисунок 7.3.1 – Кількість вільних місць у палаті 3

Функція повертає основну інформацію про останній прийом клієнта, id якого передано параметром.

drop function if exists get\_appointment\_info;

delimiter $$

create function get\_appointment\_info(client\_id int) returns varchar(255)

deterministic

begin

declare appointment\_info varchar(255);

select concat('Service: ',service\_name,', Date: ', appointment\_date,', Price: ', coalesce(service.price, 0)) into appointment\_info

from appointment

join service on appointment.service\_id = service.id

where appointment.patient\_id = client\_id

order by appointment\_date desc

limit 1;

return coalesce(appointment\_info, 'This patient has no appointments');

end $$

delimiter ;

select get\_appointment\_info(3);



Рисунок 7.3.2 – Основна інформація про останній прийом клієнта з id = 3

## Створення представлень

Представлення показує загальну інформацію про пацієнтів.

drop view if exists patient\_data;  
create or replace view patient\_data as  
select patient.id, patient\_name, sex, height, weight, blood\_type,  
 allergies, medical\_history  
from patient  
join medical\_card mc on patient.id = mc.patient\_id;  
select \* from patient\_data;

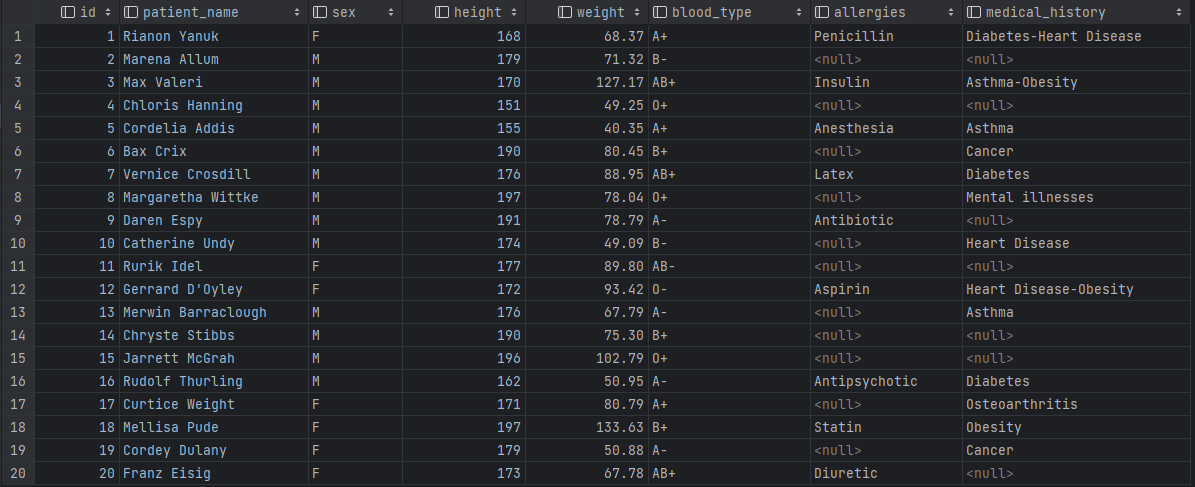


Рисунок 7.4.1 – Представлення з загальною інформацією про пацієнтів

Представлення показує загальну інформацію про прийоми.

drop view if exists appointment\_data;

create or replace view appointment\_data as

select appointment.id,appointment\_date,doctor\_name, speciality,

patient\_name, service\_name, duration\_minutes, price

from appointment

join patient p on appointment.patient\_id = p.id

join doctor d on appointment.doctor\_id = d.id

join service s on appointment.service\_id = s.id

order by appointment\_date;

select \* from appointment\_data;

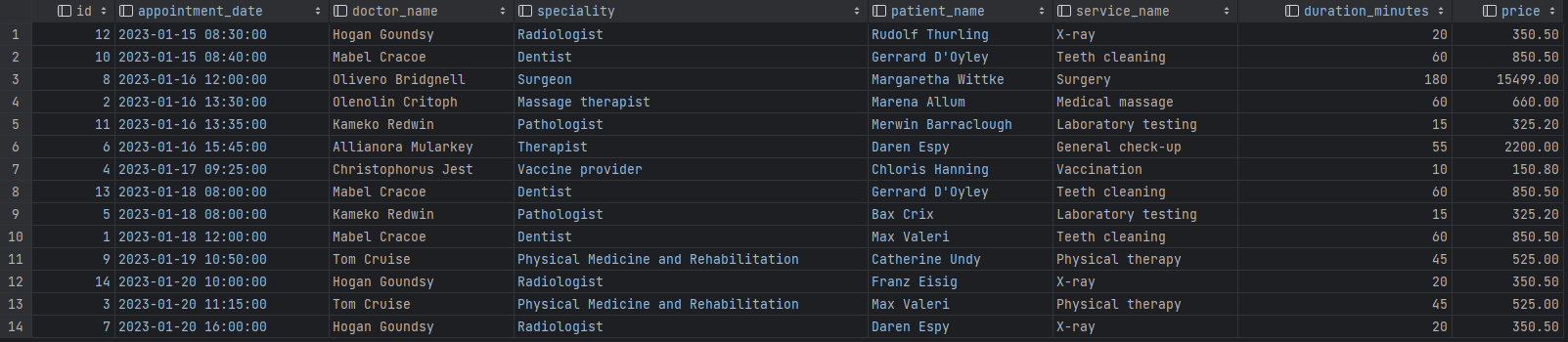


Рисунок 7.4.2 – Представлення з загальною інформацією про прийоми

## Створення різних запитів

Запит показує пацієнтів та інформацію про палати, де вони лежать.

select end\_date, patient\_name, room\_number, type

from room\_housing

join patient p on room\_housing.patient\_id = p.id

join room r on room\_housing.room\_id = r.id

where curdate() between start\_date and end\_date

order by room\_number;

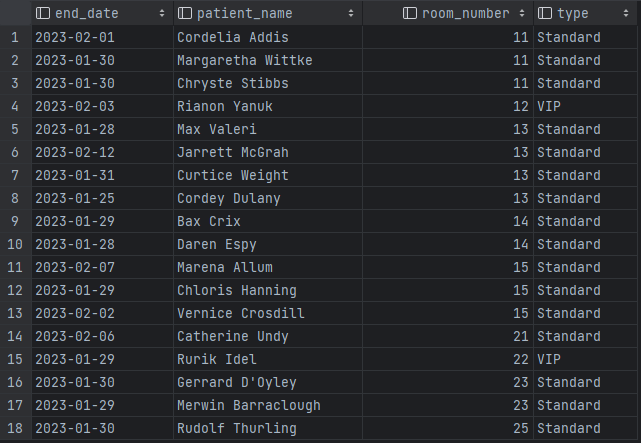


Рисунок 7.5.1 – Пацієнти та палати, в яких вони лежать

Запит показує загальну суму оплати та її метод, здійсненої кожним пацієнтом.

select patient.id, patient\_name, payment\_method,

sum(amount) as amount\_of\_payments

from patient

left join appointment a on patient.id = a.patient\_id

left join payment p on a.payment\_id = p.id

group by patient.id, payment\_method

order by amount\_of\_payments desc;

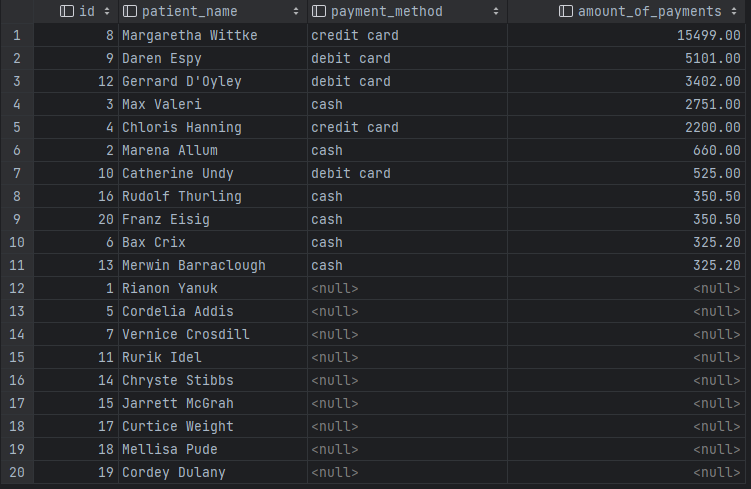


Рисунок 7.5.2 – Сума оплата кожного пацієнта

Запит показує всіх клієнтів та інформацію про медикаменти, що виписані їм.

select patient.id, patient\_name, medication\_name, dosage\_in\_mg, price

from patient

left join appointment a on patient.id = a.patient\_id

left join medication\_appointment ma on a.id = ma.appointment\_id

left join medication m on ma.medication\_id = m.id;

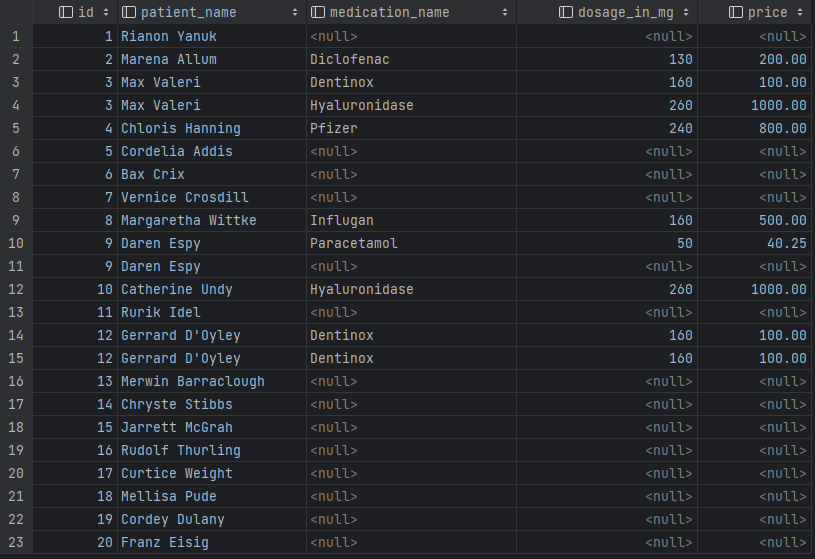


Рисунок 7.5.3 – Пацієнти та їх медикаменти

Запит показує усіх пацієнтів, що мають медичне страхування.

select patient.id, patient\_name

from patient

join medical\_card on patient.id = medical\_card.patient\_id

where medical\_card.availability\_of\_insurance = 1;



Рисунок 7.5.4 – Усі пацієнти з медичним страхуванням

Запит показує усі прийоми, здійснені протягом останнього тижня.

select service\_name, doctor\_name, appointment\_date

from appointment

join service on appointment.service\_id = service.id

join doctor on appointment.doctor\_id = doctor.id

where appointment\_date between DATE\_ADD(curdate(), INTERVAL -7 DAY) and curdate()

order by appointment\_date;

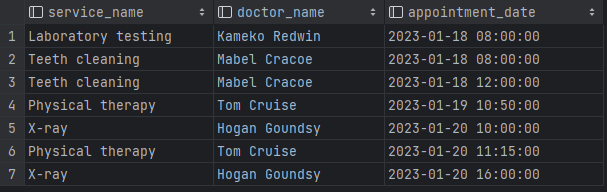


Рисунок 7.5.5 – Усі прийоми за останній тиждень

Запит показує кількість прийомів, здійснених кожним лікарем, та загальну суму грошей, що були зароблені на цих прийомах.

select doctor.id, doctor\_name, count(\*) as appointments\_amount,

sum(price) as earned\_money, service\_name

from doctor

join appointment a on doctor.id = a.doctor\_id

join service s on a.service\_id = s.id

group by doctor.id, service\_name

order by earned\_money desc;

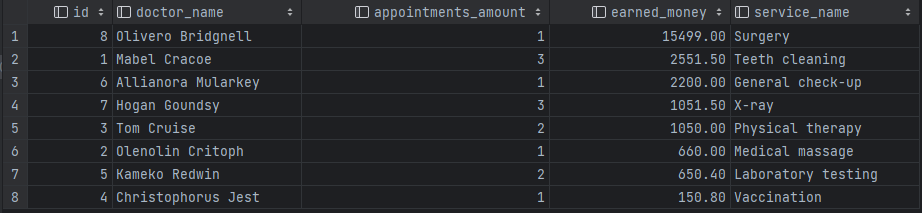


Рисунок 7.5.6 – Усі лікарі, їх кількість прийомів та зароблених грошей за них

Запит показує кількість пацієнтів в кожній палаті на даний момент.

select room\_number, type, count(\*) as patients\_amount

from room\_housing

join room r on room\_housing.room\_id = r.id

group by room\_number, type;

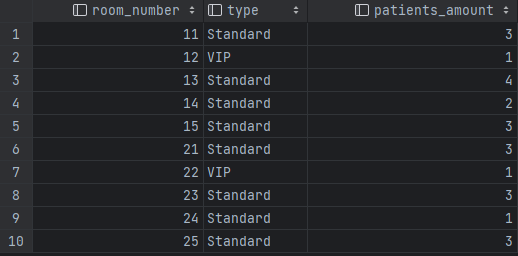


Рисунок 7.5.7 – Усі палати та кількість мешканців в даний момент

Запит показує медикаменти та кількість разів, що вони були виписані пацієнтам.

select medication\_name, dosage\_in\_mg, count(\*) as prescription

from medication

join medication\_appointment ma on medication.id = ma.medication\_id

group by medication\_name, dosage\_in\_mg;

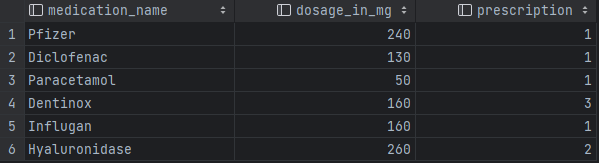


Рисунок 7.5.8 – Ліки та кількість разів, що вони були виписані

Запит показує кількість медикаментів, які потрібно замовити (виписані пацієнтам, але нема в наявності).

select medication\_name, count(\*) as required\_amount

from medication\_appointment

join medication m on m.id = medication\_appointment.medication\_id

where availability = 0

group by medication\_name;

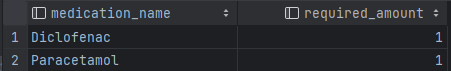


Рисунок 7.5.9 – Кількість невистачаючих медикаментів

Запит показує кількість унікальних пацієнтів у кожного лікаря.

select doctor.id, doctor\_name, speciality,

count(distinct patient\_id) as unique\_patients

from doctor

left join appointment a on doctor.id = a.doctor\_id

group by doctor.id, speciality;

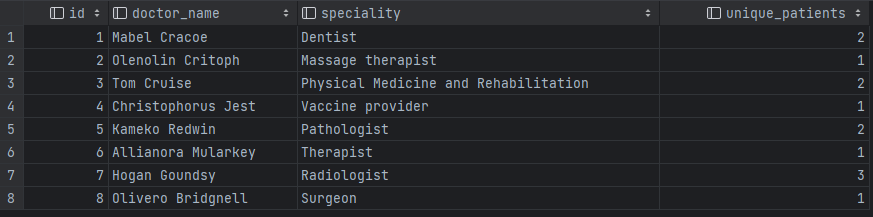


Рисунок 7.5.10 – Лікарі та кількість їх пацієнтів

Запит показує пацієнтів, що не мають алергій та жодних записів у історії хвороб.

select patient\_name

from patient

join medical\_card mc on patient.id = mc.patient\_id

where allergies is null and medical\_history is null;

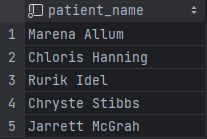


Рисунок 7.5.11 – Усі пацієнти без алергій та записів у історії хвороб

Запит показує усіх пацієнтів, що мають четверту позитивну групу крові та можуть бути універсальними реципієнтами.

select patient\_name

from patient

join medical\_card mc on patient.id = mc.patient\_id

where blood\_type = 'AB+';

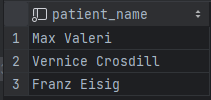


Рисунок 7.5.12 – Усі пацієнти з четвертою позитивною гурпою крові

Запит показує звіт по цінам до кожного прийому, включаючи вартість виписаних медикаментів та самої послуги

select appointment.id, coalesce(m.price, 0) as med\_price, s.price as service\_price,

(select (sum(med\_price + service\_price))) as general\_price

from appointment

left join service s on appointment.service\_id = s.id

left join medication\_appointment ma on appointment.id = ma.appointment\_id

left join medication m on ma.medication\_id = m.id

order by id;



Рисунок 7.5.13 – Звіт про ціни до кожного прийому

Запит показує середню вартість одного прийому, мінімальну та максимальну ціну прийому та загальну вартість усіх прийомів, враховуючи медикаменти та послуги, що надаються.

select round(avg(coalesce(m.price, 0) + s.price),2) as avg\_price,

max(coalesce(m.price, 0) + s.price) as max\_price,

min(coalesce(m.price, 0) + s.price) as min\_price,

sum(coalesce(m.price, 0) + s.price) as sum\_price

from appointment

left join service s on appointment.service\_id = s.id

left join medication\_appointment ma on appointment.id = ma.appointment\_id

left join medication m on ma.medication\_id = m.id;



Рисунок 7.5.14 – Звіт з середньою, максимальною, мінімальною вартістю прийому та суму вартостей усіх прийомів

Запит показує усіх пацієнтів, що мають критичний індекс маси тіла.

select patient\_name, height, weight, weight/(height\*height/10000) as BMI

from patient

join medical\_card mc on patient.id = mc.patient\_id

where not weight/(height\*height/10000) between 16.5 and 30;

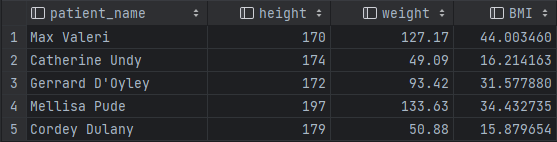


Рисунок 7.5.15 – Пацієнти з критичним ІМТ

Запит показує кількість та загальну суму оплат кожного виду.

select payment\_method, count(\*) as payments\_amount, sum(amount) as payments\_sum

from payment

group by payment\_method;

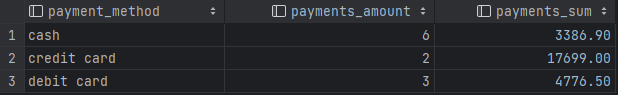


Рисунок 7.5.16 – Звіт по методами оплати

Запит показує часові межі кожного прийому та їх тривалість.

select appointment.id, appointment\_date, duration\_minutes,

date\_add(appointment\_date, interval duration\_minutes minute) as end\_datetime

from appointment

join service s on appointment.service\_id = s.id;

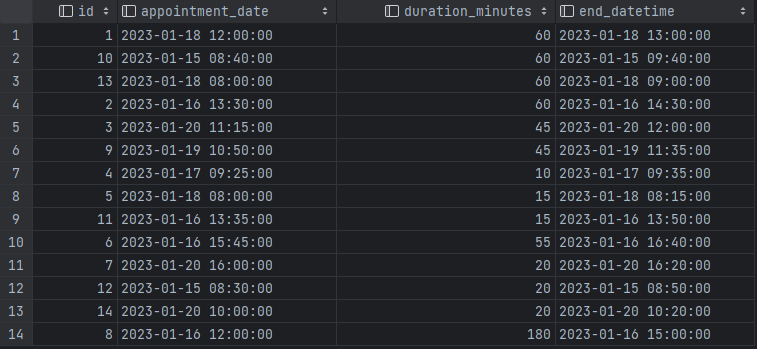


Рисунок 7.5.17 – Часові межі кожного прийому

Запит показує історію хвороб та наявність страхування у пацієнтів уразливого віку (60+).

select patient.id, patient\_name, medical\_history,

date\_format(from\_days(datediff(now(),date\_of\_birth)), '%Y')+0 AS age,

availability\_of\_insurance

from patient

join medical\_card mc on patient.id = mc.patient\_id

having age > 60;

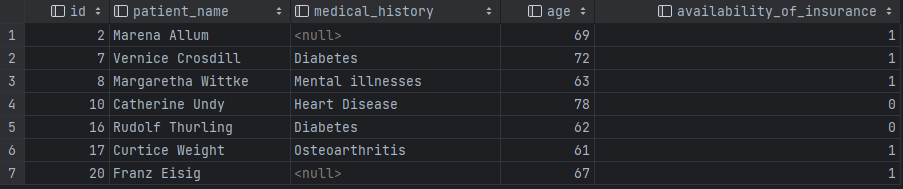


Рисунок 7.5.18 – Пацієнти уразливого віку

## Створення індексів

Продемонструємо роботу індексів на прикладі нижче. Для початку виконаємо звичайний запит, з якого дістанемо дані про записи в палаті, зроблені 30.12.2022.

explain

select \*

from room\_housing

where start\_date = '2022-12-30';

На рисунку нижче можна побачити, що загальний час виконання запиту становив 60 мс.



Рисунок 7.6.1 – Аналіз виконання запиту

Далі створимо індекс та повторимо той самий запит ще раз.

create index test

on service(price);

Отримаємо результат в 42 мс, що значно швидше, ніж попереднє виконання цього ж запиту.



Рисунок 7.6.2 – Аналіз виконання запиту

З отриманих результатів можна наглядно побачити, як індекси оптимізують роботу запитів та значно збільшують швидкість їх виконання.

Висновок

Щоб розробити комплексну базу даних лікувального закладу, спочатку був розроблений комплексний план, було визначено ключові суб’єкти та їх відносини, а також встановлено основні бізнес-правила та положення. Завдяки цим зусиллям було успішно створено та впроваджено повнофункціональну базу даних. Загалом було створено 10 окремих таблиць, які пов’язані разом за допомогою зовнішніх ключів. Крім того, було продемонстровано досвід у створенні та застосуванні різних елементів бази даних, таких як тригери, функції, процедури та складні запити, які оптимізовано за допомогою індексів.

Перелік посилань

1. https://dev.mysql.com/doc/
2. <https://www.jetbrains.com/help/datagrip/meet-the-product.html>
3. https://www.w3schools.com/mysql/default.asp