**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ**

**УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-**

**аналітики**

**КУРСОВА РОБОТА**

**з навчальної дисципліни**

**“Проектування та адміністрування БД і СД”**

**на тему:**

**«Інформаційна система діяльності приватної лікарні»**

**спеціальність:** 051 «Економіка»

**спеціалізація:** «Інформаційні технології в бізнесі»

**освітній ступінь:** бакалавр

**Науковий керівник: Виконавець:**

к.ф.-м.н., доц. Депутат Б.Я.Шматюк А.І.

(прізвище, ім’я, по-батькові) (прізвище, ім’я, по-батькові)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) УФЕ-31с група

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

**“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 р. “\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2022 р.**

**Загальна кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підписи, ПІП членів комісії)

**Львів 2022**

ЗМІСТ

[ВСТУП………………………………………………………………………….…3](#_Toc104409504)

[РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ………………………………....6](#_Toc104409505)

[1.1 Опис предметної області та її бізнес процеси…………………………....6](#_Toc104409506)

[1.2 Аналіз вимог для інформаційної системи………………………………...8](#_Toc104409507)

[1.3 Use case diagram…………………………………………………………...10](#_Toc104409508)

[РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ…………………………………………14](#_Toc104409509)

[2.1 Система управління базами даних MySQL 14](#_Toc104409510)

[2.2 Опис моделі даних 16](#_Toc104409511)

[2.3 Визначення типів даних 18](#_Toc104409512)

[2.5 Обмеження цілісності даних; 20](#_Toc104409513)

[ВИСНОВКИ……………………………………………………………………...29](#_Toc104409514)

[СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………….30](#_Toc104409515)

[ДОДАТКИ………………………………………………………………………..31](#_Toc104409516)

# ВСТУП

**Актуальність теми дослідження.**  Інформаційна система — це формальна соціотехнічна організаційна система, призначена для збору, обробки, зберігання та поширення інформації. З соціально-технічної точки зору, інформаційні системи складаються з чотирьох компонентів: завдання, люди, структура (або ролі) і технологія.

Інформаційна система охорони здоров’я дає змогу організаціям охорони здоров’я збирати, зберігати, керувати, аналізувати й оптимізувати історію лікування пацієнтів та інші ключові дані. Ці системи також дозволяють постачальникам медичних послуг легко отримувати інформацію про макросередовища, наприклад про тенденції здоров’я в громаді. Доступ до інформаційних систем охорони здоров’я відіграє важливу роль у стриманні витрат. Лікарі можуть аналізувати історію пацієнтів, а також подібні тематичні дослідження та приймати більш точні рішення щодо лікування. Тим часом адміністратори можуть контролювати кількість пацієнтів у різних відділах і розумно розподіляти персонал та інші ресурси там, де вони найбільше потрібні.

Сучасна лікарня відрізняється від тих, які були років п'ятнадцять або двадцять тому. І перше, що кидається в очі, відвідавши її, ви не знайдете там черг, які можна було спостерігати раніше. Це пов'язано з тим, що з'явилося багато приватних лікарень і кабінетів, готових на високому якісному рівні приймати пацієнтів, які потребують послуг різних лікарів. А так само це пов'язано з появою електронних черг. Тому зараз більшість установ з обслуговування населення переходять на електронну чергу, виходячи з цього, ідея спростити реєстрацію, збір даних про клієнтів і запис їх до фахівця носить актуальний характер. За допомогою інформаційних систем охорони здоров’я організації можуть краще конкурувати та процвітати в медичній сфері, що все більше цифровізується. Вони мають кращі можливості для досягнення цілей охорони здоров’я якомога ефективніше, а також надають лікарям і медсестрам можливість досягти найкращих результатів для своїх пацієнтів.

Лікарня - це така організація, яка працює з дуже великим обсягом інформації, як про співробітників, так і про пацієнтів. Лікарям завжди стежити за даними про своїх пацієнтів, про курс лікування хворих. А керівництву та бухгалтерії необхідно бути в курсі подій про своїх співробітників. Мета інформаційної системи діяльності приватної лікарні — інтегрувати, збирати, зберігати та керувати даними з ряду джерел для підтримки оперативного управління охороною здоров’я, підтримки політичних рішень та керування даними пацієнтів. Для цього потрібна спільна база даних, що включає всю необхідну інформацію.

**Мета і завдання дослідження.** *Метою курсової роботи є* розробка та аналіз бази даних діяльності приватної лікарні.

Для досягнення мети в роботі поставлено й вирішено такі теоретичні та практичні завдання:

* коротко охарактеризувати інформаційну систему для приватної поліклініки;
* проаналізувати предметну область;
* визначити бізнес-процеси та вимоги до інформаційної системи приватної лікарні;
* розробити use case diagram для інформаційної системи приватної лікарні;
* розробити архітектуру програмної системи та спроектувати структуру

бази даних;

**Об’єктом дослідження**курсової роботи є особливість розробки інформаційної системи для приватної поліклініки за допомогою системи управління базами даних MYSQL.

**Предметом дослідження** є діяльність приватної лікарні, що спеціалізується різних захворюваннях.

В процесі розробки використовуватиметься мова програмування та технології – MySQL.

**Структура роботи.** Курсова робота складається з 2 розділів ( «Аналіз предметної області», «Розробка бази даних»») висновків, списку використаних джерел та додатків. Загальний обсяг роботи − 46 сторінок.

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області та її бізнес процеси

Приватна лікарня – це медичний заклад, який має своїх фахівців і займається прийомом пацієнтів за попереднім записом. Інформація про кожного пацієнта зберігається в базі даних лікарні, тому є можливість простежити за всіма відвідуваннями обраного пацієнта даної клініки. Також інформаційна система дозволяє простежити шлях пацієнта від реєстратури до прийому у певного лікаря, у певному кабінеті та за певною послугою. Оплата прийому здійснюється після отримання амбулаторної картки, яка оформляється в реєстратурі. В клієнта може бути знижка, яка теж занесена в базу. Система призначена для обробки даних про лікарів, пацієнтів, прийомі пацієнтів, повинна видавати звіти за запитом адміністрації.

Перед прийомом до лікарні пацієнт проходить реєстрацію. Якщо пацієнт вже був на прийомі у даній лікарні, тоді проводиться пошук по базі і даного пацієнта записують на прийом і видають амбулаторний лист. У амбулаторному аркуші зазначається в на який день призначено прийом, до якого фахівця (спеціалізація, прізвище лікаря), в якому кабінеті здійснюватиметься прийом, вартість приймання. Якщо ж пацієнт вперше в даній лікарні, тоді йому присвоюється унікальний реєстраційний номер, заносяться в базу даних (прізвище, ім'я та по батькові, дата народження, пошта і знижка, якщо вона є), а потім вже відбувається запис на прийом до конкретного лікаря і на конкретну дату. Пацієнт повинен бути зареєстрований в системі до прийому в лікарні. Один пацієнт може бути записаний на прийом до декількох лікарів та може використовувати декілька послуг. Після реєстрації пацієнта, в призначений день він приходить на прийом до фахівця. Лікар здійснює пошук і вибір даного пацієнта, вибирає прийом, і виходячи з обраного прийому, робить запис про діагноз і призначення лікування даному пацієнтові. При повторному прийомі відбувається редагування історії хвороби.

Бізнес-процес – це сукупність бізнес-завдань і дій, які, коли виконуються людьми або системами в структурованому курсі, дають результат, який сприяє досягненню бізнес-цілей. У цій статті ми навчаємо визначення бізнес-процесів і розглянемо його переваги, приклади та пов’язані з ними бізнес-терміни. Процеси є скрізь у бізнесі. Від залучення нових співробітників до співбесід, виставлення рахунків постачальникам і продажів, кожен бізнес-результат є результатом повторюваних завдань і дій.

Важливість бізнес-процесів в предметній області неможливо переоцінити. Оптимізація бізнес-процесів може допомогти уникнути помилок і затримок, покращуючи економічну ефективність, продуктивність, відносини з постачальниками та задоволеність клієнтів.

В предметній області виділено наступні бізнес-процеси (рис 1.1.1) :

* здійснення прийомів;
* облік отриманих сум (оплата прийомів);
* облік фахівців та клієнтів;

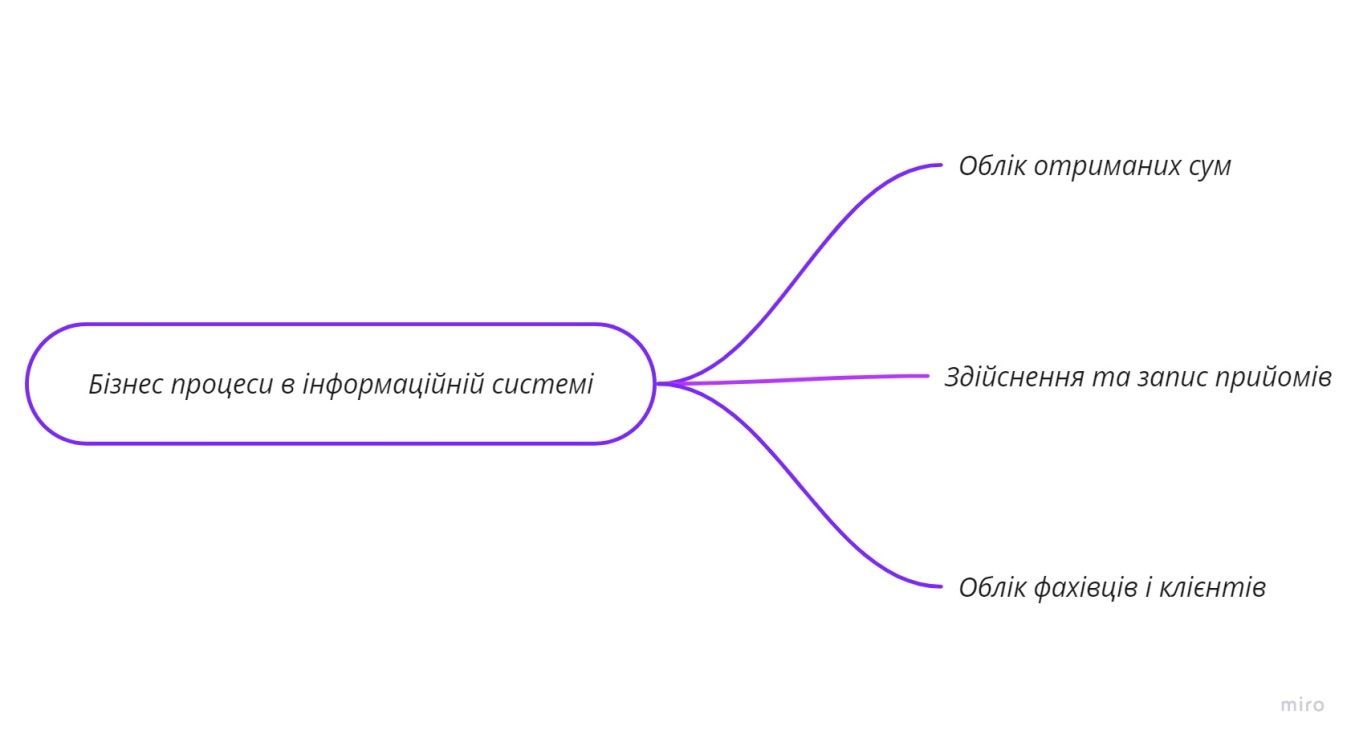


Рис. 1.1.1 Діаграма бізнес-процесів

Необхідність і переваги бізнес-процесу цілком очевидні у великих організаціях. Процес формує рятівний круг для будь-якого бізнесу та допомагає йому впорядкувати окремі види діяльності, забезпечуючи оптимальне використання ресурсів.

## 1.2 Аналіз вимог для інформаційної системи

Вимоги бізнесу визначають, що організація хоче або має бути в змозі зробити після завершення проекту. Вони описують зміни можливостей, які будуть наслідком проекту. Технічні вимоги, з іншого боку, визначають рішення щодо того, як буде задоволена потреба кожного проекту. Зазвичай вони включають вимоги щодо брендингу, досвіду клієнтів, управління ризиками, інформаційної безпеки, операцій, обслуговування, відповідності та зручності використання. Зазвичай нефункціональні вимоги посилаються на зовнішні документи, такі як стандарти, політики та процедури.

Важливим аспектом в аналізі вимог до проекту є SWOT-аналіз. SWOT означає сильні сторони, слабкі сторони, можливості та загрози, тому SWOT-аналіз є технікою для оцінки цих чотирьох аспектів вашого бізнесу. SWOT-аналіз – це інструмент, який може допомогти вам проаналізувати, що ваша компанія робить найкраще зараз, і розробити успішну стратегію на майбутнє.

Для SWOT-аналізу візьмемо дві інформаційні системи приватних лікарень: Добробут і Medikom. Розглянемо сильні сторони, слабкі сторони, можливості та загрози.

Першим проаналізуємо Добробут.

Сильні сторони:

* дуже широко описаний спектр послуг;
* доступно висвітлено демократичні ціни;
* робота в режимі 24/7;
* діджитал-медицина в додатку;

Слабкі сторони:

* недостатній список власних лікарів;
* деяка інформація про обладнання потребує оновлення;

Можливості:

* розробка системи лояльності для постійних клієнтів;

Загрози:

* висока конкуренція на ринку;
* залежність від закордонних постачальників;

В ІС приватної лікарні «Добробут» присутні і плюси, і мінуси. Але, сильні стороний все ж таки переважають.

Проаналізуємо Medikom.

Сильні сторони:

* широкий спектр послуг;
* робота в режимі 24/7;
* висвітлено високі медичні стандарти та професіоналізм;
* програма лояльності для постійних клієнтів;

Слабкі сторони:

* не висвітлено ціни за послуги, що не дає можливості аналізу для клієнта ;
* інформація про розташування не зрозуміла;

Можливості:

* удосконалення інформаційної системи;
* діджитал-медицина в додатку;

Загрози:

* висока конкуренція на ринку;
* перехід клієнтів в інші лікарні через незрозумілість в ІС;

В ІС приватної лікарні «Medikom» є більше мінусів. ІС потребує допрацювання.

Отже, зробивши SWOT-аналіз, можна сказати, що головне в ІС це доступність, як в висвітленні послуг, так і цін на них. Система лояльності клієнтів є обов’язковим критерієм. Широкий спектр наповненості і лікарів, і послуг надає змогу клієнту впевнитися у рейтингу лікарні. Також, важливим є доступність такої інформації, як розташування і графік роботи.

## 1.3 Use case diagram

Use case diagram— це спосіб узагальнити деталі системи та користувачів у цій системі. Загалом це показано як графічне зображення взаємодій між різними елементами в системі. Діаграми варіантів використання вказують події в системі та те, як ці події протікають, однак діаграма варіантів використання не описує, як ці події реалізуються. Наявність діаграми варіантів використання має ряд переваг перед подібними діаграмами, такими як блок-схеми.

Причини, чому організації повинні використовувати діаграми випадків, включають:

* представлення цілей систем і користувачів;
* контекст, у якому система має розглядатися;
* формування системних вимог;
* формується модель для потоку подій, коли йдеться про взаємодію користувачів;
* зовнішній вигляд системи;
* зовнішні та внутрішні впливи на системи;

Тому, розробка use case diagram є дуже важливим етапом в створенні інформаційної системи. Так, як за допомогою цієї діаграми, можна сформувати модель взаємодії користувачів та адміністраторів з системою.

Use case diagram для інформаційної системи діяльності приватної лікарні подана на рис. 1.3.1:

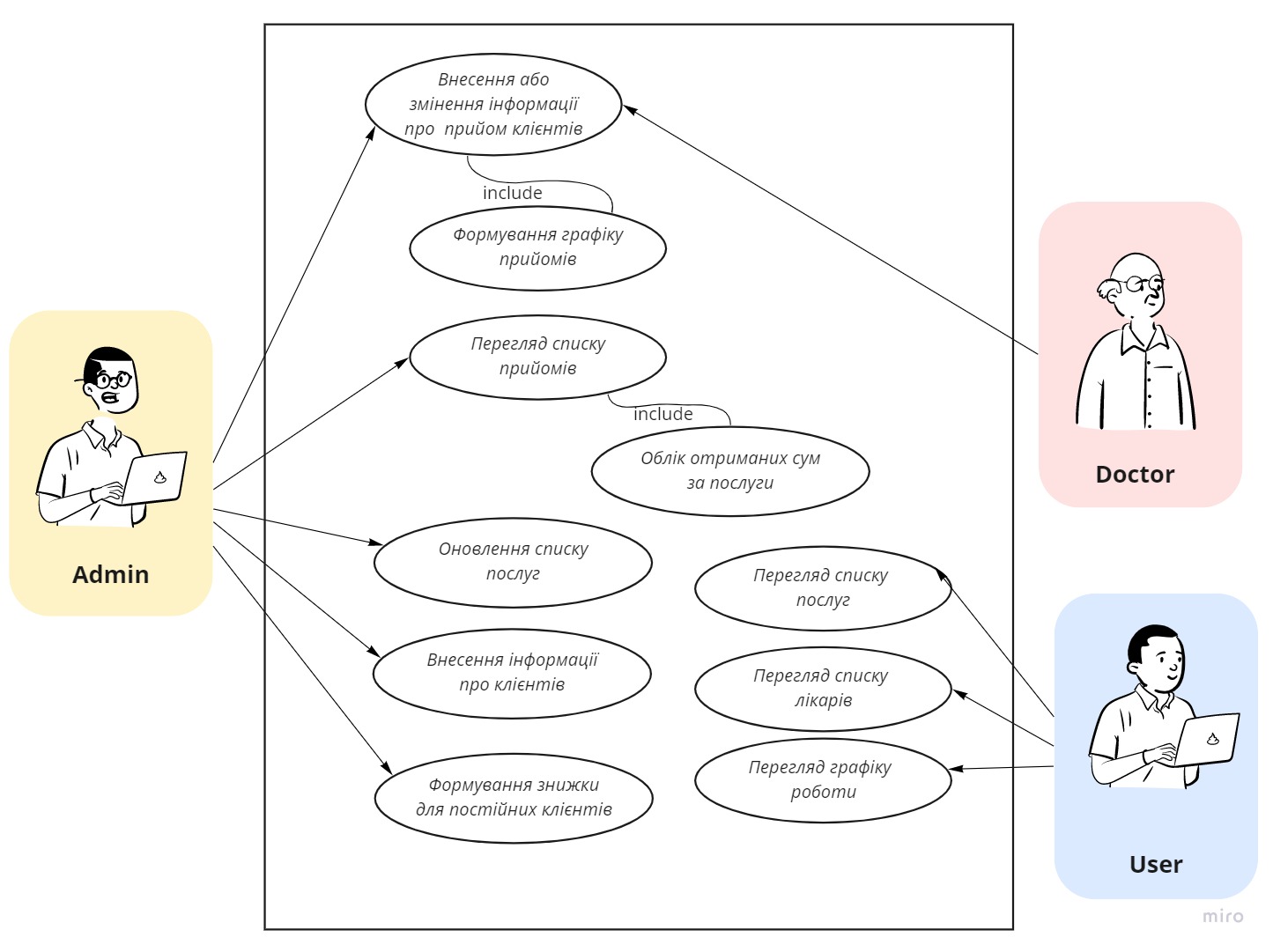


Рис 1.3.1 Use case diagram інформаційної системи діяльності приватної лікарні

З діаграми ми можемо бачити, що є 3 актори, які взаємодіють з системою:

* admin ( адміністратор) – це людина, яка керує системою та має найбільше функцій, а саме:
* вносить або змінює інформацію про прийом клієнтів –адміністратор може записувати клієнтів на прийом, вносити інформацію, що стосується прийому, змінювати записи;
* складає графік прийомів – адміністратор може відслідковувати коли той чи інший лікар зайнятий або ні і записувати клієнта на прийом;
* переглядає список прийомів – адміністратор може переглядати інформацію про прийом для аналізу та збору інформації;
* переглядає список роботи лікарів – може переглядати коли лікарі працюють, а коли ні, щоб надавати інформацію клієнтам;
* веде облік отриманих сум – здійснює підрахунок сум, які лікарня отримала за послуги, в ті чи інші дні;
* формує знижку для постійних клієнтів – здійснює аналітику інформації про клієнтів, які відвідують лікарню та можуть мати знижку, через постійне відвідування чи кількість послуг, якими вони користуються
* оновлює список послуг – адміністратор може оновлювати послуги, які є в лікарні, добавляти нові або вносити зміни;
* doctor ( лікар) – це працівник лікарні і людина, яка використовує систему та має такі функцій, а саме:
* вносить або змінює інформацію про прийом клієнтів – лікар вносити інформацію, що стосується прийому певного клієнта;
* формує графік прийомів – може вносити свої корективи в графік прийомів;
* переглядає список прийомів – може відслідковувати коли той чи інший пацієнт записаний на прийом для формування своєї роботи;
* user ( клієнт) – це людина, яка використовує систему в своїх цілях та має такі функцій, а саме:
* переглядає список послуг – клієнт ознайомлюється з списком послуг та ціною на них;
* переглядає список лікарів – клієнт ознайомлюється з списком лікарів та аналізує який йому підходить найбільше;
* переглядає графік роботи – клієнт ознайомлюється з графіком роботи, щоб в подальшому знати, коли він може прийти в лікарню;

Отже, розробка use case diagram є дуже важливим та корисним фактором. Роблячи висновки та аналіз, ми можемо побудувати досконалу та функціональну інформаційну систему, яка взаємодіє з користувачами.

# РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ

## 2.1 Система управління базами даних MySQL

SQL - проста мова програмування, яка має небагато команд і якій може навчитися будь-який бажаючий. Розшифровується як Structured Query Language - мова структурованих запитів, яка була розроблена для роботи з БД, а саме, щоб отримувати/добавляти/змінювати дані, мати можливість опрацьовувати великі масиви інформації та швидко отримувати структуровану та згруповану інформацію. Є багато варіантів мови SQL, але у них всіх основні команди майже однакові. Також існує і багато СУБД, але основними з них являються: Microsoft Access, Microsoft SQL Server, MySQL, Oracle SQL, IBM DB2 SQL, PostgreSQL та Sybase Adaptive Server SQL.

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та зберігаються в одному місці. Якщо коротко, то найпростіша БД це звичайна таблиця з рядками та стовпцями у якій зберігається різного роду інформація (прикладом може слугувати таблиця в Excel). Так, часто, з БД нероздільно пов'язують Системи управління базами даних (СУБД), які надають функціонал для роботи з БД. Мова SQL якраз і є частиною СУБД, яка здійснює керування інформацією в БД. Ми будемо вважати БД набором звичайних таблиць, які зберігаються в окремих файлах. Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

* кожен елемент таблиці - один елемент даних;
* всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);
* кожен стовпчик має унікальне ім'я;
* однакові рядки в таблиці відсутні;
* порядок проходження рядків і стовпчиків може бути довільним.

Виділяють три різновиди зв'язку між таблицями бази даних:

* «один-до-багатьох»;
* «один-до-одного»;
* «багато-до-багатьох».

*Зв'язок «один-до-багатьох»*

Зв'язок «один-до-багатьох» має місце, коли одному запису батьківської таблиці може відповідати декілька записів дочірньої. Зв'язок «один-до-багатьох» іноді називають зв'язком «багато-до-одного». І у тому, і в іншому випадку суть зв'язку між таблицями залишається незмінною. Зв'язок «один-до-багатьох» є найпоширенішим для реляційних баз даних. Вона дозволяє моделювати також ієрархічні структури даних.

*Зв'язок «один-до-одного»*

Зв'язок «один-до-одного» має місце, коли одному запису в батьківській таблиці відповідає один запис в дочірній. Це відношення зустрічається набагато рідше, ніж відношення «один-до-багатьох». Його використовують, якщо не хочуть, щоб таблиця БД «розпухала» від другорядної інформації, проте для читання зв'язаної інформації в декількох таблицях доводиться проводити ряд операцій читання замість однієї, коли дані зберігаються в одній таблиці.

Зв'язок «багато-до-багатьох» застосовується в наступних випадках:

одному запису в батьківській таблиці відповідає більш за один запис в дочірній, або одному запису в дочірній таблиці відповідає більш за один запис в батьківській.

Будь-який зв'язок «багато-до-багатьох» в реляційній базі даних необхідно замінити на зв'язок «один-до-багатьох» (одну або більш) за допомогою введення додаткових таблиць.

Переваги SQL:

* може використовуватися на різних платформах.
* робить швидку оцінку запитів.
* може використовуватися для роботи з великими масивами.
* знаходиться під ліцензією GPL (OpenSource).
* має підтримку від великої спільноти користувачів / розробників.

Недоліки MySQL:

Незважаючи на широке застосування, MySQL також має недоліки, і зручність користування дуже залежить від області, в якій вона застосовується.

* обмежена функціональність у порівнянні з великими системами управління базами даних.
* дорога ліцензія для комерційного використання.

## 2.2 Опис моделі даних

В процесі проектування структури бази даних потрібно створити діаграму концептуальної моделі даних. На основі визначених елементів і зв’язків створити ER – діаграму(рис 2.1). У процесі розробки проекту у базі даних «med» створено такі таблиці:

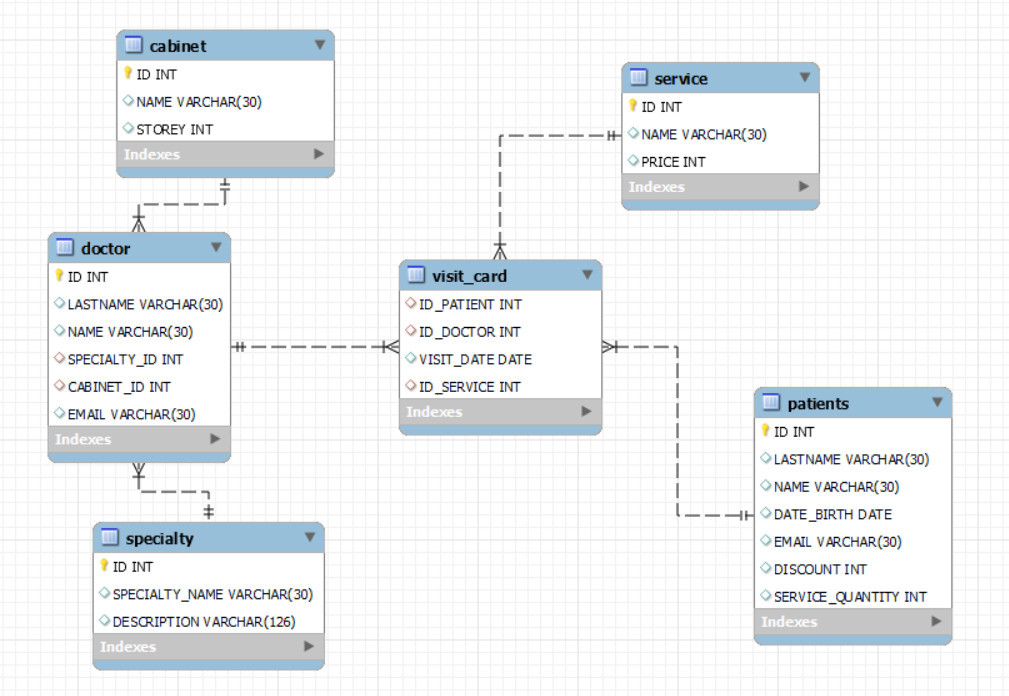


Рис. 2.1. ER – діаграма

* doctor – таблиця, яка містить інформацію про лікарів в медичному закладі;
  + id – унікальний номер;
  + lastname – прізвище лікаря;
  + name – ім’я лікаря;
  + specialty\_id – вторинний ключ, описує спеціалізацію;
  + cabinet\_id – вторинний ключ, описує кабінет;
  + email – пошта лікаря;
* cabinet – таблиця, яка містить інформацію про кабінет та на якому поверсі він розташований;
* id – унікальний номер;
* name – назва кабінету;
* storey – поверх, на якому знаходиться кабінет;
* specialty – таблиця, яка містить інформацію про спеціалізацію;
* id – унікальний номер;
* specialty\_name – назва спеціалізації;
* description – опис, чим займається та чи інша спеціалізація;
* service – таблиця, яка містить інформацію про послуги в лікарні;
* id – унікальний номер;
* name – назва послуги;
* price – ціна послуги;
* patients
* id – унікальний номер;
* lastname – прізвище пацієнта;
* name – ім’я пацієнта;
* date\_birth – дата народження пацієнта;
* email – пошта пацієнта;
* discount – знижка;
* service\_quantity – кількість послуг, якими скористався пацієнт;
* visit\_card – таблиця, яка містить інфомацію про візит клієнта до лікарні;
* id\_patient – вторинний ключ, номер пацієнта;
* id\_doctor – вторинний ключ, номер лікаря;
* visit\_date – дата візиту;
* id\_service – вторинний ключ, номер послуги;

## 2.3 Визначення типів даних

Типи даних бази даних відносяться до формату зберігання даних, який може містити окремий тип або діапазон значень. Коли комп’ютерні програми зберігають дані у змінних, кожна змінна повинна мати окремий тип даних. Деякі поширені типи даних: цілі числа, символи, рядки, числа з плаваючою комою та масиви.

Типи даних в таблиці “doctor”:

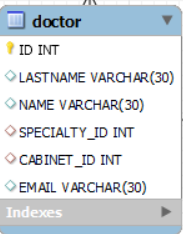


Рис 2.2. Таблиця «doctor»

Типи даних в таблиці «speciality»:

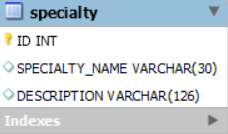


Рис 2.2. Таблиця «speciality»

Типи даних в таблиці «cabinet»:

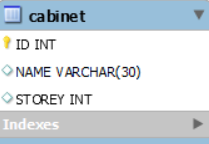


Рис 2.2. Таблиця «cabinet»

Типи даних в таблиці «service»:

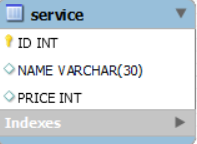


Рис 2.2. Таблиця «service»

Типи даних в таблиці «patients»:

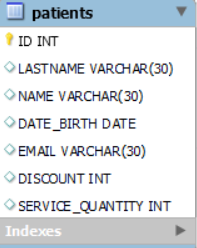


Рис 2.2. Таблиця «patients»

Типи даних в таблиці «visit \_card»:

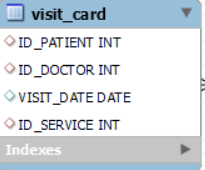


Рис 2.2. Таблиця «visit\_card»

Основні типи даних, які використовувались:

VARCHAR – може зберігати не більше 255 символів. На відміну від CHAR, для зберігання значення даного типу виділяється необхідна кількість пам'яті;

INT – діапазон від -2 147 483 648 до 2 147 483 647;

## 2.5 Обмеження цілісності даних;

Правила цілісності бази даних - це правила, які дають змогу уникнути введення некоректних даних у БД, а також забезпечити можливість зв'язування декількох таблиць. Ці правила можуть бути описані при створенні чи модифікації таблиці.

Типи правил цілісності:

* CHECK - Контроль допустимих значень атрибутів.
* NOT NULL/NULL - Заборона/ дозвіл на використання не заданих або не визначених значень.
* UNIQUE - Контроль унікальності значень атрибутів.
* PRIMARY KEY - Первинний ключ.
* FOREIGN KEY - Зовнішній ключ.

FOREIGN KEY  використовується для того, щоб зв'язати дві таблиці, в нього є такі особливості :

* поле у головній таблиці, на яке буде адресуваться FG повинне бути оголошене PK або UNIQUE.
* поле у головній таблиці і поле з якої відбувається адресація повинні мати один й теж самий тип даних
* також за допомогою FG можна реалізувати каскадне видалення і оновлення даних.

PRIMARY KEY використовується для ідентифікації рядків таблиці, в нього є певні особливості:

* в одній таблиці БД може бути тільки один PK.
* рядки, які мають це правило, не можуть мати не визначені або не задані дані.
* будучи призначеним для декількох стовпців(складений ключ), це обмеження ставить унікальність комбінацій відповідних значень, хоча окреме значення в кожному стовпці складеного ключа не обов'язково має бути унікальним.

Обмеження UNIQUE (унікальний), як і обмеження PRIMARY KEY, обмежує множину значень для вказаних стовпців унікальними значеннями. Як і обмеження PRIMARY KEY, обмеження UNIQUE може бути обмеженням таблиці, і тоді воно визначає унікальність комбінацій значень відповідних стовпців.

Обмеження NULL / NOT NULL дозволяють або забороняють введення в поле NULL-значень. Очевидно, що обмеження NOT NULL має бути вказано для первинних ключів, оскільки в іншому випадку під загрозою опиниться цілісність даних. Зв’язки між таблицями в базі даних подано на рис.2.5.1.

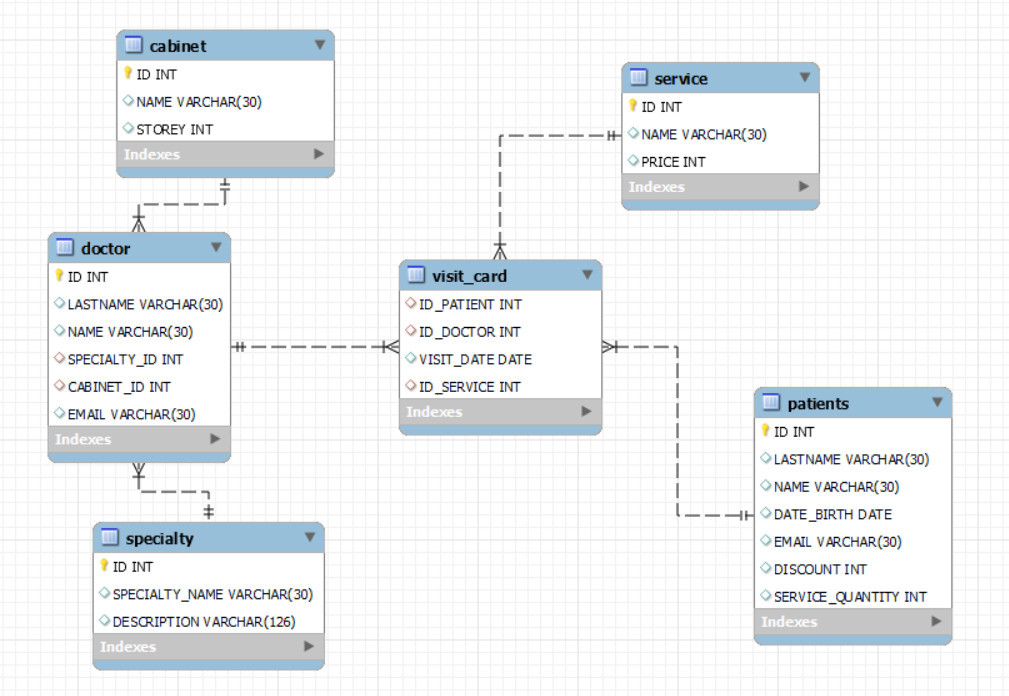


Рис. 2.5.1. Демонстрація зв’язків між таблицями в базі даних

При роздобці даної бази даних використовувалось:

* primary key, foreign key.

**2.6 Реалізація SQL-скрипту**

SQL скрипт — це набір команд SQL, збережених у вигляді файлу в SQL Scripts. Скрипт SQL може містити один або кілька операторів SQL або блоків PL/SQL. Ви можете використовувати скрипт SQL для створення, редагування, перегляду, запуску та видалення файлів.

Запити типу select:

Запит який виводить ціну за знижкою (якщо така є) яку платив пацієнт в певного лікаря за певну послугу. Реалізує функцію обліку сум для адміністратора. Результат зображено на рис.2.6.1:

SELECT

PATIENTS.NAME AS PATIENT\_NAME,

PATIENTS.LASTNAME AS PATIENT\_LASTNAME,

DOCTOR.NAME AS DOCTOR\_NAME,

DOCTOR.LASTNAME AS DOCTOR\_LASTNAME,

SERVICE.PRICE - PATIENTS.DISCOUNT AS PRICE,

SERVICE.NAME AS SERVICE\_NAME

FROM PATIENTS

JOIN VISIT\_CARD

ON ID = VISIT\_CARD.ID\_PATIENT

JOIN SERVICE

ON SERVICE.ID = VISIT\_CARD.ID\_SERVICE

JOIN DOCTOR

ON DOCTOR.ID = VISIT\_CARD.ID\_DOCTOR

WHERE DOCTOR.NAME = 'Igor' AND SERVICE.NAME = 'gastro uzd'

SELECT \*FROM VISIT\_CARD WHERE visit\_card.VISIT\_DATE BETWEEN '2022-03-27' and now();

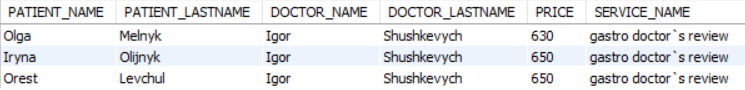


Рис. 2.6.1

Запит який виводить Ім'я, Прізвище та електронну пошту докторів, які мають певну спеціальність також виводить опис цієї спеціальності. Реалізує функцію перегляду списку прийомів для адміністратора. Результат зображено на рис.2.6.2:

SELECT

DOCTOR.NAME,

DOCTOR.LASTNAME,

DOCTOR.EMAIL,

SPECIALTY.SPECIALTY\_NAME,

SPECIALTY.DESCRIPTION

FROM DOCTOR

JOIN SPECIALTY

ON DOCTOR.SPECIALTY\_ID = SPECIALTY.ID

WHERE SPECIALTY.SPECIALTY\_NAME = 'family doctor'

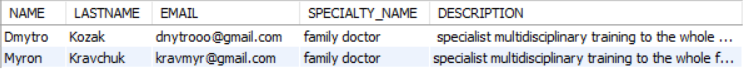


Рис. 2.6.2

Запит який виводить клієнтів, які платили більше 750грн. Реалізує функцію формування знижки для клієнтів для адміністратора. Результат зображено на рис.2.6.3:

SELECT

PATIENTS.NAME,

PATIENTS.LASTNAME,

PATIENTS.EMAIL

FROM PATIENTS

JOIN VISIT\_CARD

ON PATIENTS.ID = VISIT\_CARD.ID\_PATIENT

JOIN SERVICE

ON VISIT\_CARD.ID\_SERVICE = SERVICE.ID

WHERE SERVICE.PRICE >= 750

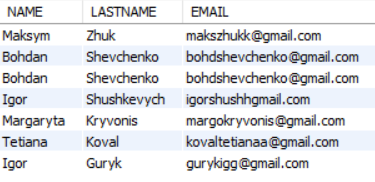


Рис. 2.6.3

Запит який виводить інформацію про лікарів, кабінети та поверхи, які відвідував певний пацієнт. Реалізує функцію перегляду списку прийомів для адміністратора. Результат зображено на рис.2.6.4:

SELECT

DOCTOR.NAME AS DOCTOR\_NAME,

DOCTOR.LASTNAME AS DOCTOR\_LASTNAME,

DOCTOR.EMAIL AS DOCTOR\_EMAIL,

CABINET.NAME AS CABINET\_NAME,

CABINET.STOREY AS CABINET\_STOREY,

PATIENTS.NAME AS PATIENT\_NAME,

PATIENTS.LASTNAME AS PATIENT\_LASTNAME

FROM PATIENTS

JOIN VISIT\_CARD

ON PATIENTS.ID = VISIT\_CARD.ID\_PATIENT

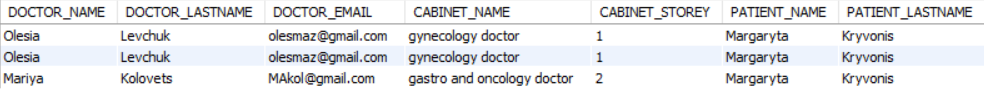
JOIN DOCTOR

ON VISIT\_CARD.ID\_DOCTOR = DOCTOR.ID

JOIN CABINET

ON DOCTOR.CABINET\_ID = CABINET.ID

WHERE PATIENTS.NAME = 'Margaryta' AND PATIENTS.LASTNAME = 'Kryvonis

Рис 2.6.4

Запит який виводить Ім'я, Прізвище, електронну пошту пацієнта та дату відвідування (в порядку зростання). Реалізує функцію формування графіків прийомів для адміністратора. Результат зображено на рис.2.6.5:

SELECT

VISIT\_CARD.VISIT\_DATE,

PATIENTS.NAME,

PATIENTS.LASTNAME,

PATIENTS.EMAIL

FROM VISIT\_CARD

JOIN PATIENTS

ON VISIT\_CARD.ID\_PATIENT = PATIENTS.ID

ORDER BY(VISIT\_DATE);

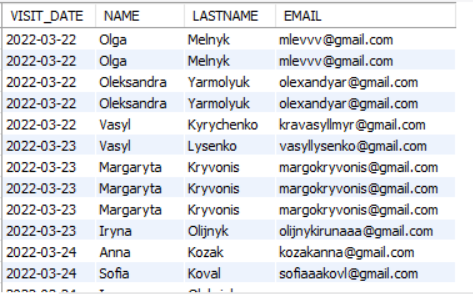


Рис 2.6.5

Заповнення таблиць. ). Реалізує функцію внесення даних для адміністратора. Результат на рис. 2.6.6:

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES ( 1, 'Yaskiv','Anton', 2, 2, 'YAANT@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 2, 'Shushkevych', 'Igor', 1, 3, 'igg1or@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 3, 'Barna', 'Anna', 3, 5, 'barnaann@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 4, 'Polishna','Tetiana', 5, 5, 'polishna@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 5, 'Kozak', 'Dmytro', 7, 1, 'dnytrooo@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 6, 'Shevchenko', 'Petro', 6, 1, 'petttro@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES ( 7, 'Levchuk', 'Olesia', 4, 4, 'olesmaz@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 8, 'Kravchuk', 'Myron', 9, 1, 'kravmyr@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 9, 'Kolovets', 'Mariya', 10, 3, 'MAkol@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 10,'Semenyuk', 'Andrij', 8, 2, 'semenukanriii@gmail.com');

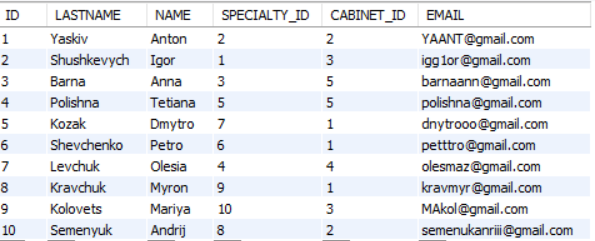


Рис. 2.6.6

Таким чином в базі даних заповняються всі інші таблиці. Та проходить оновлення інформації про пацієнтів, послуги, записи на прийом.

# ВИСНОВКИ

Організації охорони здоров’я постійно генерують дані. Сюди входять дані про обсяг роботи, день перебування в лікарні, тенденції щодо здоров’я пацієнтів, виставлення рахунків, витрати та доходи, тощо. Основна мета інформаційних систем охорони здоров’я полягає в тому, щоб допомогти організаціям отримати ці дані, інтерпретувати їх і застосувати на практиці. Отже, розробка такої ІС проходила в кілька етапів.

На першому етапі даної роботи, було проаналізовано предметну область даної системи, визначено бізнес-процеси та вимоги до інформаційної системи приватної лікарні та виконаний SWOT-аналіз систем на ринку. Також, було розроблено та описано use case діаграму.

На другому етапі було описаносистему управління базами даних MySQL. Спроектовано та описано структуру БД, її таблиці та записи. Визначено типи даних та обмеження цілісності даних. Для кращого розуміння архітектури продукту було розроблено діаграми, які показують взаємодію користувачів та їх функцій між собою та у системі. Реалізовано SQL-скрипт в якому наведено заповнення даних та запити, які випливають з use case діаграми.

Таким чином, розроблена база може бути використана для збільшення ефективності роботи лікарні шляхом прискорення таких операцій, як запис пацієнтів на прийом, ведення первинної фінансової звітності, створення та зберігання архівної інформації про пацієнтів в зручній для пошуку та обробки формі.

# СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пасічник В. В. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко. – К.: Видавнича група BHV, 2017. – 384 с.
2. Діго С.М. Бази даних. Проектування та створення. Підручник. 2010-210с.
3. Глушаков, С.В Ломодько Д.В Бази даних/ Навчальний посібник 2009р.-156c.
4. Автоматизовані інформаційні технології в економіці: Підручник/За ред. проф. Г.А. Титоренко, 2005. –399 с
5. Калашян О.М., Калянов Г.М. Структурні моделі бізнесу: DFD-технології. М.: Фінанси і статистика, 2003. – 256 с.
6. Федотова Д.Е. Семенов Ю.Д., Чижик К.Н. CASE-технології: Практикум. М.: Гаряча лінія-Телеком, 2005. – 160 с.
7. Роб П., Коронел К. Системи баз даних: проектування, реалізація та управління, 2004. – 1040 с.
8. Data Base [Електронний ресурс]. – Режим доступу : <https://www.oracle.com/database/what-is-database/>
9. Igi Global [Електронний ресурс]. – Режим доступу : <https://www.igi-global.com/dictionary/hospital-information-system-his/13245>
10. Kiss flow [Електронний ресурс]. – Режим доступу : <https://kissflow.com/workflow/bpm/business-process/>
11. Miro [Електронний ресурс]. – Режим доступу : <https://miro.com/app/dashboard/>
12. Вікіпедія. [Електронний ресурс]. - Режим доступу: <https://uk.wikipedia.org>
13. W3schools [Електронний ресурс]. - Режим доступу: <https://www.w3schools.com/sql/sql_datatypes.asp>

# ДОДАТКИ

Додаток А

**Створення та заповнення бази даних**

CREATE DATABASE MED;

USE MED;

CREATE TABLE DOCTOR ( ID INT PRIMARY KEY,

LASTNAME VARCHAR (30),

NAME VARCHAR (30),

SPECIALTY\_ID INT,

CABINET\_ID INT,

EMAIL VARCHAR (30));

CREATE TABLE SPECIALTY ( ID INT PRIMARY KEY,

SPECIALTY\_NAME VARCHAR (30),

DESCRIPTION VARCHAR (126) );

CREATE TABLE CABINET ( ID INT PRIMARY KEY,

NAME VARCHAR (30),

STOREY INT );

CREATE TABLE SERVICE ( ID INT PRIMARY KEY,

NAME VARCHAR (30),

PRICE INT );

CREATE TABLE PATIENTS ( ID INT PRIMARY KEY,

LASTNAME VARCHAR (30),

NAME VARCHAR (30),

DATE\_BIRTH DATE,

EMAIL VARCHAR (30),

DISCOUNT INT,

SERVICE\_QUANTITY INT);

CREATE TABLE VISIT\_CARD ( ID\_PATIENT INT,

ID\_DOCTOR INT,

VISIT\_DATE DATE,

ID\_SERVICE INT);

ALTER TABLE DOCTOR ADD FOREIGN KEY(SPECIALTY\_ID) REFERENCES SPECIALTY(ID);

ALTER TABLE DOCTOR ADD FOREIGN KEY(CABINET\_ID) REFERENCES CABINET(ID);

ALTER TABLE VISIT\_CARD ADD FOREIGN KEY(ID\_SERVICE) REFERENCES SERVICE (ID);

ALTER TABLE VISIT\_CARD ADD FOREIGN KEY(ID\_DOCTOR) REFERENCES DOCTOR (ID);

ALTER TABLE VISIT\_CARD ADD FOREIGN KEY(ID\_PATIENT) REFERENCES PATIENTS (ID);

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 1, 'gastroenterology', 'disease of the digestive tract');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 2, 'cardiology', 'heart diseases');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 3, 'otology', 'ear diseases');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 4, 'gynevology', 'about women`s health');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 5, 'dermatologists', 'skin disease');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 6, 'pediatricians', 'treat children and their illnes');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 7, 'family doctor', ' specialist multidisciplinary training to the whole family');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 8, 'neurologist', 'disease of the brain and spinal cord, nerves');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 9, 'family doctor', 'specialist multidisciplinary training to the whole family');

INSERT INTO SPECIALTY ( ID , SPECIALTY\_NAME , DESCRIPTION )

VALUES ( 10, 'oncology', 'diagnosis and treatment of benign and malignant neoplasms') ;

INSERT INTO CABINET ( ID , NAME , STOREY)

VALUES (1, 'family and pediatry doctor', 1);

INSERT INTO CABINET ( ID , NAME , STOREY)

VALUES (2, 'cardio and neurology doctor', 2);

INSERT INTO CABINET ( ID , NAME , STOREY)

VALUES (3, 'gastro and oncology doctor', 2);

INSERT INTO CABINET ( ID , NAME , STOREY)

VALUES (4, 'gynecology doctor', 1);

INSERT INTO CABINET ( ID , NAME , STOREY)

VALUES (5, 'dermo and otology doctor', 1);

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES ( 1, 'Yaskiv','Anton', 2, 2, 'YAANT@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 2, 'Shushkevych', 'Igor', 1, 3, 'igg1or@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 3, 'Barna', 'Anna', 3, 5, 'barnaann@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 4, 'Polishna','Tetiana', 5, 5, 'polishna@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 5, 'Kozak', 'Dmytro', 7, 1, 'dnytrooo@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 6, 'Shevchenko', 'Petro', 6, 1, 'petttro@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES ( 7, 'Levchuk', 'Olesia', 4, 4, 'olesmaz@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 8, 'Kravchuk', 'Myron', 9, 1, 'kravmyr@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 9, 'Kolovets', 'Mariya', 10, 3, 'MAkol@gmail.com');

INSERT INTO DOCTOR ( ID , LASTNAME , NAME , SPECIALTY\_ID , CABINET\_ID , EMAIL )

VALUES( 10,'Semenyuk', 'Andrij', 8, 2, 'semenukanriii@gmail.com');

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES (1, 'Melnyk', 'Olga', '1999-03-15', 'mlevvv@gmail.com', 20, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 2, 'Lysenko', 'Vasyl', '1989-12-22', 'vasyllysenko@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 3, 'Yarmolyuk', 'Oleksandra', '1999-02-22', 'olexandyar@gmail.com', 0, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 4, 'Kozak', 'Anna', '1998-02-16', 'kozakanna@gmail.com', 20, 1);

INSERT INTO PATIENTS( ID , LASTNAME , NAME , DATE\_BIRTH, EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 5, 'Koval', 'Sofia', '1997-03-15', 'sofiaaakovl@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 6, 'Shevchenko', 'Taras', '1992-06-26', 'shevtarrr@gmail.com', 0, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES (7, 'Shushkevych', 'Igor', '1997-04-24', 'igorshushhgmail.com', 50, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH, EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 8, 'Kyrychenko', 'Vasyl', '1989-02-22', 'kravasyllmyr@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 9, 'Kryvonis', 'Margaryta', '1991-08-26', 'margokryvonis@gmail.com', 10, 3);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 10,'Olijnyk', 'Iryna', '2001-07-15', 'olijnykirunaaa@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 11,'Koval', 'Tetiana', '2000-01-19', 'kovaltetianaa@gmail.com', 10, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 12,'Zhuk', 'Maksym', '1997-10-15', 'makszhukk@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 13,'Levchul', 'Orest', '2003-03-08', 'levchukorestt@gmail.com', 0, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 14,'Lysenko', 'Markijan', '1996-04-15', 'marklysenkoo@gmail.com', 10, 3);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 15,'Goncharenko', 'Anna', '1992-03-30', 'gonchanna@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 16,'Shmatiuk', 'Maksym', '1996-04-15', 'maksshmatiukkk@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 17,'Shmatiuk', 'Olexandr', '2006-01-13', '', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 18,'Oleksiuk', 'Iruna', '2007-05-19', '', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 19, 'Guryk', 'Igor', '1974-06-07', 'gurykigg@gmail.com', 0, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 20, 'Kravchuk', 'Tetiana', '1974-07-28', 'kravchuktettt@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 21, 'Kravchuk', 'Nastya', '2008-10-09', '', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 22, 'Nesteruk', 'Viktor', '2002-10-22', 'victornest@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 23, 'Martyniuk', 'Iruna', '2000-08-07', 'martyniukiruna@gmail.com', 0, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 24, 'Martyniuk', 'Volodumur', '1999-05-12', 'martyniukvolod@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 25, 'Kravets', 'Inna', '1986-09-24', 'innakravets@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 26, 'Koval', 'Oleg', '1983-11-22', 'kovalolegg@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 27, 'Shevchenko', 'Bohdan', '1991-03-17', 'bohdshevchenko@gmail.com', 10, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 28, 'Shmatiuk', 'Orest', '2019-07-10', '', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 29, 'Shmatiuk', 'Tetiana', '1997-06-26', 'tetianashmatiukk@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 30, 'Levchuk', 'Victoria', '1984-08-21', 'levchukvikttt@gmail.com', 10, 2);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 31, 'Levutska', 'Victoria', '1999-07-08', 'levutvictoryy@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 32, 'Lusenko', 'Anton', '1989-11-05', 'antonlysenko@gmail.com', 0, 3);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 33, 'Kovalchuk', 'Oleg', '1995-01-15', 'kovalchuk@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 34, 'Zhuk', 'Anastasiya', '2002-12-09', 'zhuknastyaaa@gmail.com', 0, 1);

INSERT INTO PATIENTS ( ID , LASTNAME , NAME , DATE\_BIRTH , EMAIL, DISCOUNT, SERVICE\_QUANTITY )

VALUES( 35, 'Kyrychok', 'Evgenii', '2001-04-01', 'kkkyrychok@gmail.com', 10, 2);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 1, 'gastro doctor`s review', 650);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 2, 'gastro uzd', 450);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 3, 'cardio doctor`s review', 700);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 4, 'cardiograma ', 500);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 5, 'cardio uzd ', 450);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 6, 'otology doctor`s review ', 400);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 7, 'gynecology doctor`s review ', 600);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 8, 'gynecology uzd ', 450);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 9, 'dermatology doctor`s review', 650);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 10, 'dermatology analizes', 400);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 11, 'pediatry doctor`s review', 600);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 12, 'family doctor`s review', 600);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 13, 'neurology doctor`s review', 750);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 14, 'neurology eeg ', 700);

INSERT INTO SERVICE ( ID , NAME, PRICE)

VALUES ( 15, 'oncology doctor`s review', 850);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 1, 2, '2022-03-22' , 1);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 1, 2, '2022-03-22' , 2);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 2, 4, '2022-03-23' , 9);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 3, 4, '2022-03-22' , 9);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 3, 4, '2022-03-22' , 10);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 4, 6, '2022-03-24' , 11);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 5, 5, '2022-03-24' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 6, 1, '2022-03-25' , 4);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 6, 1, '2022-03-25' , 5);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 7, 9, '2022-03-25' , 15);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 8, 3, '2022-03-22' , 6);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 9, 7, '2022-03-23' , 7);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 9, 7, '2022-03-23' , 8);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 9, 9, '2022-03-23' , 15);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 10, 2, '2022-03-23' , 1);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 11, 9, '2022-03-26' , 15);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 12, 10, '2022-03-30' , 13);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 13, 2, '2022-03-28' , 1);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 13, 2, '2022-03-28' , 2);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 14, 1, '2022-03-29' , 3);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 14, 1, '2022-03-29' , 4);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 14, 1, '2022-03-29' , 5);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 15, 10, '2022-03-27' , 14);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 16, 3, '2022-03-27' , 9);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 17, 6, '2022-03-24' , 11);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 18, 6, '2022-03-24' , 11);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 19, 9, '2022-03-26' , 15);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 19, 4, '2022-03-26' , 10);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 20, 7, '2022-04-01' , 7);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 21, 6, '2022-04-01' , 11);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 22, 5, '2022-04-01' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 23, 1, '2022-04-03' , 3);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 23, 1, '2022-04-03' , 5);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 24, 5, '2022-04-05' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 25, 5, '2022-04-05' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 26, 10, '2022-04-05' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 27, 10, '2022-04-05' , 14);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 27, 10, '2022-04-05' , 13);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 28, 6, '2022-04-03' , 11);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 29, 7, '2022-04-03' , 7);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 30, 7, '2022-04-01' , 7);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 30, 7, '2022-04-01' , 8);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 31, 5, '2022-04-05' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 32, 1, '2022-04-05' , 3);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 32, 1, '2022-04-05' , 4);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 32, 1, '2022-04-05' , 5);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 33, 5, '2022-04-07' , 12);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 34, 3, '2022-04-07' , 6);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 35, 1, '2022-04-07' , 3);

INSERT INTO VISIT\_CARD ( ID\_PATIENT, ID\_DOCTOR, VISIT\_DATE, ID\_SERVICE )

VALUES ( 35, 1, '2022-04-07' , 6);

Додаток Б

**Запити для бази даних**

-Яка послуга була використана найбільше разів та яка найменше---

SELECT

SERVICE.NAME AS SERVICE\_NAME,

COUNT(SERVICE.NAME) AS VALUE\_NAME

FROM VISIT\_CARD

JOIN SERVICE

ON SERVICE.ID = VISIT\_CARD.ID\_SERVICE

GROUP BY SERVICE.NAME

ORDER BY VALUE\_NAME DESC

-- Запит який виводить кількість відвідувачів та загальну суму за вибрану дату-------

SELECT VISIT\_DATE AS DATE\_OF\_VISIT,

COUNT(DISTINCT ID\_PATIENT) AS COUNT\_OF\_PATIENT,

SUM(PRICE) AS TOTAL\_PRICE

FROM VISIT\_CARD

RIGHT JOIN SERVICE ON ID=VISIT\_CARD.ID\_SERVICE

WHERE VISIT\_DATE = '2022-04-05'

GROUP BY VISIT\_DATE;

-- До якого саме лікаря (family doctor)з двох ходять на прийом більше людей

SELECT

DOCTOR.NAME AS DOCTOR\_NAME,

DOCTOR.LASTNAME AS DOCTOR\_LASTNAME,

COUNT(VISIT\_CARD.ID\_PATIENT) AS COUNT\_OF\_PATIENT

FROM VISIT\_CARD

JOIN DOCTOR

ON VISIT\_CARD.ID\_DOCTOR = DOCTOR.ID

JOIN SPECIALTY

ON DOCTOR.SPECIALTY\_ID = SPECIALTY.ID

WHERE SPECIALTY\_NAME = 'family doctor'

GROUP BY DOCTOR.NAME, DOCTOR.LASTNAME

ORDER BY DOCTOR.NAME, DOCTOR.LASTNAME DESC

LIMIT 1;

-- Запит який виводить ціну за знижкою (якщо така є) яку платив пацієнт в певного лікаря за певну послугу

SELECT

PATIENTS.NAME AS PATIENT\_NAME,

PATIENTS.LASTNAME AS PATIENT\_LASTNAME,

DOCTOR.NAME AS DOCTOR\_NAME,

DOCTOR.LASTNAME AS DOCTOR\_LASTNAME,

SERVICE.PRICE - PATIENTS.DISCOUNT AS PRICE,

SERVICE.NAME AS SERVICE\_NAME

FROM PATIENTS

JOIN VISIT\_CARD

ON ID = VISIT\_CARD.ID\_PATIENT

JOIN SERVICE

ON SERVICE.ID = VISIT\_CARD.ID\_SERVICE

JOIN DOCTOR

ON DOCTOR.ID = VISIT\_CARD.ID\_DOCTOR

WHERE DOCTOR.NAME = 'Igor' AND SERVICE.NAME = 'gastro doctor`s review';

--Клієнти, які платили більше 750грн

SELECT

PATIENTS.NAME,

PATIENTS.LASTNAME,

PATIENTS.EMAIL

FROM PATIENTS

JOIN VISIT\_CARD

ON PATIENTS.ID = VISIT\_CARD.ID\_PATIENT

JOIN SERVICE

ON VISIT\_CARD.ID\_SERVICE = SERVICE.ID

WHERE SERVICE.PRICE >= 750

-- Запит який віводить всіх користувачів в яких є знижка

SELECT \* FROM PATIENTS

WHERE DISCOUNT > 0;

-- Запит який виводить інформацію про лікарів (або лікаря) та на якому поверсі вони знаходяться, в певному кабінеті

SELECT

DOCTOR.NAME,

DOCTOR.LASTNAME,

SPECIALTY.SPECIALTY\_NAME,

CABINET.NAME,

CABINET.STOREY

FROM DOCTOR

JOIN SPECIALTY

ON DOCTOR.SPECIALTY\_ID = SPECIALTY.ID

JOIN CABINET

ON DOCTOR.CABINET\_ID = CABINET.ID

WHERE CABINET.NAME = 'dermo and otology doctor';

---Лікарі, які не мали клієнтів в певну дату

SELECT

DOCTOR.NAME,

DOCTOR.LASTNAME,

DOCTOR.EMAIL,

VISIT\_CARD.VISIT\_DATE

FROM VISIT\_CARD

JOIN DOCTOR

ON VISIT\_CARD.ID\_DOCTOR = DOCTOR.ID

WHERE NOT VISIT\_CARD.VISIT\_DATE = '2022-03-23'

GROUP BY DOCTOR.NAME, DOCTOR.LASTNAME, DOCTOR.EMAIL, VISIT\_CARD.VISIT\_DATE

-- Запит який виводить Ім'я, Прізвище та електронну пошту докторів, які мають певну спеціальність також виводить опис цієї спеціальності

SELECT

DOCTOR.NAME,

DOCTOR.LASTNAME,

DOCTOR.EMAIL,

SPECIALTY.SPECIALTY\_NAME,

SPECIALTY.DESCRIPTION

FROM DOCTOR

JOIN SPECIALTY

ON DOCTOR.SPECIALTY\_ID = SPECIALTY.ID

WHERE SPECIALTY.SPECIALTY\_NAME = 'family doctor';

-- Запит який виводить Ім'я, Прізвище, електронну пошту пацієнта та дату відвідування (в порядку зростання)

SELECT

VISIT\_CARD.VISIT\_DATE,

PATIENTS.NAME,

PATIENTS.LASTNAME,

PATIENTS.EMAIL

FROM VISIT\_CARD

JOIN PATIENTS

ON VISIT\_CARD.ID\_PATIENT = PATIENTS.ID

ORDER BY(VISIT\_DATE);

-- Запит який виводить інформацію про лікарів, кабінети та поверхи, які відвідував певний пацієнт

SELECT

DOCTOR.NAME AS DOCTOR\_NAME,

DOCTOR.LASTNAME AS DOCTOR\_LASTNAME,

DOCTOR.EMAIL AS DOCTOR\_EMAIL,

CABINET.NAME AS CABINET\_NAME,

CABINET.STOREY AS CABINET\_STOREY,

PATIENTS.NAME AS PATIENT\_NAME,

PATIENTS.LASTNAME AS PATIENT\_LASTNAME

FROM PATIENTS

JOIN VISIT\_CARD

ON PATIENTS.ID = VISIT\_CARD.ID\_PATIENT

JOIN DOCTOR

ON VISIT\_CARD.ID\_DOCTOR = DOCTOR.ID

JOIN CABINET

ON DOCTOR.CABINET\_ID = CABINET.ID

WHERE PATIENTS.NAME = 'Margaryta' AND PATIENTS.LASTNAME = 'Kryvonis';

--Клієнти які мали 2 і більше послуг

SELECT \* FROM PATIENTS

WHERE SERVICE\_QUANTITY >= 2

--Через селект зробити апдейт послуги «cardio uzd» на «cardio uzd and review»

UPDATE SERVICE

SET NAME = 'cardio uzd and review'

FROM

(SELECT \* FROM SERVICE WHERE NAME = 'cardio uzd' ) AS SELECTED

WHERE SERVICE.ID = SELECTED.ID

--поміняти кількість послуг пацієнта з 1 на 2

UPDATE PATIENTS

SET SERVICE\_QUANTITY = 2

FROM

(SELECT \* FROM PATIENTS WHERE SERVICE\_QUANTITY = 1) AS SELECTED

WHERE PATIENTS.ID = SELECTED.ID

--Селект який видає, чи можна записати клієнта на певну дату--( дата, лікар і кількість пацієнтів, що в нього було в той день і якщо більше 3 то не можна)

SELECT

VISIT\_CARD.VISIT\_DATE AS DATE,

DOCTOR.NAME AS NAME\_DOCTOR,

DOCTOR.LASTNAME AS LASTNAME\_DOCTOR,

COUNT(VISIT\_CARD.ID\_PATIENT) AS COUNT\_OF\_PATIENT,

CASE

WHEN COUNT(VISIT\_CARD.ID\_PATIENT) > 3 THEN 'Неможливо записати ще одного пацієнта! Ліміт 3 пацієнта на день!'

ELSE 'Пацієнта можливо записат'

END MESSAGE

FROM VISIT\_CARD

JOIN DOCTOR

ON VISIT\_CARD.ID\_DOCTOR = DOCTOR.ID

WHERE VISIT\_CARD.VISIT\_DATE = '2022-03-22'

GROUP BY DOCTOR.NAME, DOCTOR.LASTNAME, VISIT\_CARD.VISIT\_DATE