МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Кафедра наноелектроніки та модифікації поверхні

БАКАЛАВРСЬКА РОБОТА

зі спеціальності 153 – «Мікро- та наносистемна техніка»

на тему:

«Проектування та створення бази даних медичного закладу»

Сердитов Андрій Сергійович

Студент групи ФЕ-81
_____ А. С. Сердитов
«___»_____2022 р.

Науковий керівник
_____ доц. О. В. Ющенко
«___»_____2022 р.

Суми 2022

РЕФЕРАТ

Робота містить вступ, дослідження матеріалу, в якому розібрано що таке дані та база даних, поняття СУБД та моделі баз даних, різновиди моделей БД, побудова ER-діаграми, різновид зв’язків між сутностями один – до – одного, один – до – багатьох, багато – до – одного, багато – до – багатьох, створення та заповнення таблиць, утворення тригерів за допомогою SQL мови, ознайомлення правила техніки безпеки та охорони праці при роботі за комп’ютером, висновків та списку використаних джерел.

Звіт містить 41 сторінок, 39 рисунків та 21 літературних джерел.

В цій роботі я використовував БД офтальмологічної клініки “ЦЕНТР ЗОРУ”. Приватна офтальмологічна клініка надає послуги з діагностики, корекції та лікування зору.

Мета роботи – показати можливості, за яких стає можливим швидко оброблювати, видаляти та додавати в даному випадку медичного закладу дані за допомогою реляційної бази даних.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, РЕЛЯЦІЙНА БД, СУБД, ER-ДІАГРАМА, ОФТАЛЬМОЛОГІЧНИЙ ЦЕНТР, DATABASE BROWSER FOR SQLITE.

ЗМІСТ

ВСТУП

В нашому сьогоденні підприємства мають широкий спектр використання комп'ютерів. Це було необхідністю для промисловості як спосіб краще використовувати свої ресурси, а також спосіб охопити більшу групу потенційних клієнтів. Бази даних використовують майже де завгодно : роздрібна торгівля, банки, веб-сайти , склади тощо. Банки використовують бази даних для відстеження рахунки клієнтів, їх депозитів. Роздрібні магазини зберігають ціни, інформації про клієнтів, інформації про продажі та наявної кількості продукції. Сайти застосовують для збереження вмісту, інформації для входу та налаштувань клієнтів, а також можливість зберігати збережені дані користувача. Склади керують рівнем запасів та місцем їх зберігання . Бази даних використовуються всюди, де потрібно зберігати та легко отримувати дані.

Вперше бази даних були створені в 1960-х роках. Ранні БД були мережевими моделями, де кожен запис пов'язаний з багатьма первинними та вторинними записами. Ієрархічні бази даних також були. Вони мають деревоподібні схеми з кореневим каталогом записів, пов'язаними з декількома підкаталогами. Реляційні дані були створені в 1970-х роках. У 1980-х роках з'явилися об'єктно-орієнтовані бази даних.

У першому розділі ми дізнаємося більш детальніше о БД, їх типи, які програмні забезпечення використовувати, зв'язки між даними та нормалізація даних. У другому розглянемо на прикладі створення ER-діаграми та реляційної бази даних в СУБД для офтальмологічної клініки "ЦЕНТР ЗОРУ".

РОЗДІЛ 1 БАЗА ДАНИХ

1.1 Дані та поняття база даних.

В першу чергу треба вияснити в чому різниця між даними та інформацією.

Дані — це сукупність деталей або даних, що залишаються у формі або текстів фігур, символів, описів, або просто спостережень за сутностями, подіями чи речами, які можна проаналізувати та зробити висновки. Являються необробленими, що вимагає відтворення для отримання значущої інформації.

Інформація – дані, зіставлені для отримання значущих висновків відповідно до контекстуальних вимог. Інформація обробляється, структурується та подається з призначеним значенням, що підвищує надійність отриманих даних. Гарантує, що не залишиться невизначеності чи небажаного.

Ключові відмінності:

1. Дані містять необроблені цифри та факти, щодо інформації, надає уявлення які були проаналізовані за допомогою зібраних даних.
2. Інформація не може існувати без даних, але дані не покладаються на інформацію.
3. Дані не мають конкретики.
4. Дані не мають реального значення, тоді як інформація існує для того, щоб надати сенсу та уявлення.
5. Дані проходять процес фільтрації, за яким слідує значуща організація для створення вихідної інформації.

Термін база даних несе в собі організовану колекцію структурованої інформації або даних, взаємозв'язки між об'єктами , що підтримують одну чи декілька областей використання.

Створені для полегшення зберігання, пошуку, модифікації та видалення даних у поєднанні з різними операціями обробки даних, зазвичай БД зберігаються в комп’ютерній системі

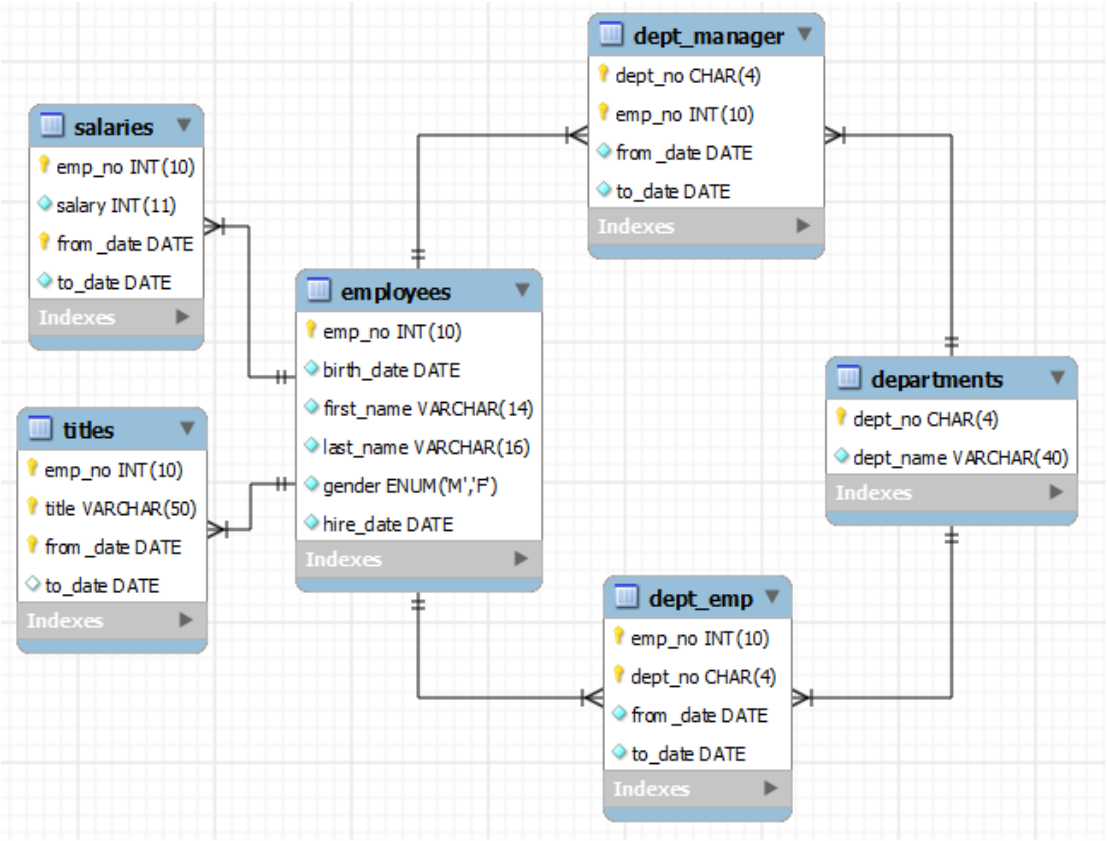

Рис. 1.1 – Модель БД[3]

1.2. Поняття СУБД, модель баз даних

Дані керуються за допомогою **системи управління базами даних(СУБД).** Система управління базами даних – це система, заснована на апаратному та програмному забезпеченні. Надає можливості збереження, створення, оновлення, маніпуляції, керування та використання баз даних. Приклади являються: DB Browser (SQLite), MySQL, Microsoft SQL Server, Oracle Database, PostgreSQL тощо. [3 - 10]

Модель бази даних або ще називають модель даних –зразок структурування даних у базі даних відповідно до формальних описів у її інформаційній системі та відповідно до вимог системи управління базою даних, яка буде застосовуватися.

Існує 3 этапи моделювання даних:

Концептуальна модель даних - ця модель визначає, що буде містити система. Включає високорівневі конструкції даних, не має технічних назв щоб на всіх рівнях могли розуміти основи даних. Може бути не нормалізованим.

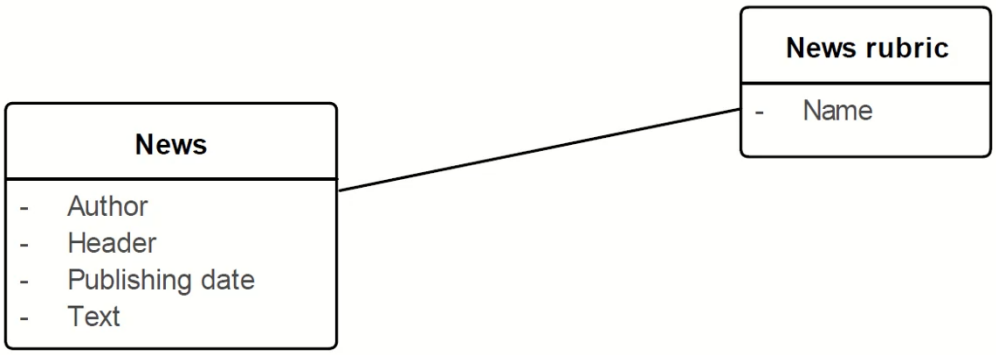

Рис. 1.2 – Концептуальна модель[5]

Логічна модель визначає, як система має бути реалізована незалежно від СУБД. Використовує в собі бізнес-назви для сутностей та атрибутів, нормалізована до четвертої нормальної форми. Метою є розроблення технічної карти правил і структур даних.

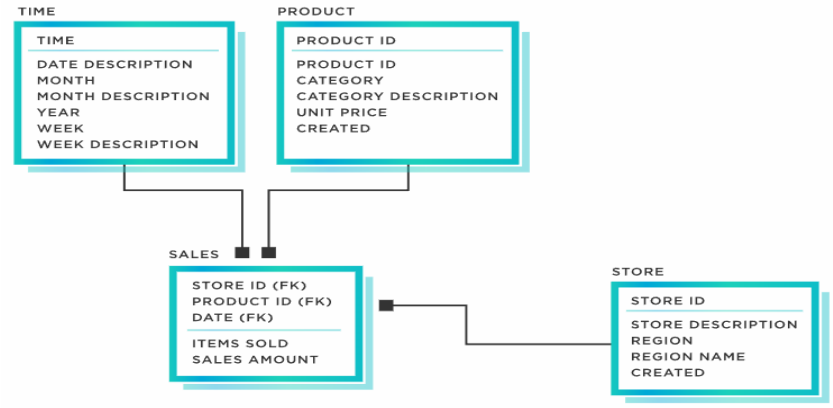

Рис. 1.3 – Логічна модель [5]

Фізична — це модель даних яка описує, як система буде реалізована за допомогою конкретної системи СУБД. Містить в собі колонки, таблиці, ключі, різні типи даних, тригери, збережені процедури, правила перевірки, обмеження доступу і домени. Використовуйте більш конкретні та менш загальні конкретні імена для стовпців і

таблиць, як скорочені імена стовпців, на які поширюються обмеження системи керування базами даних (СУБД), будь-який стандарт, визначений компанією, включаючи первинні ключі та індекси для швидкого доступу к даним. Може бути денормалізована, щоб відповідати вимогам продуктивності, заснованим на характері бази даних. Метою є фактична реалізація бази даних.

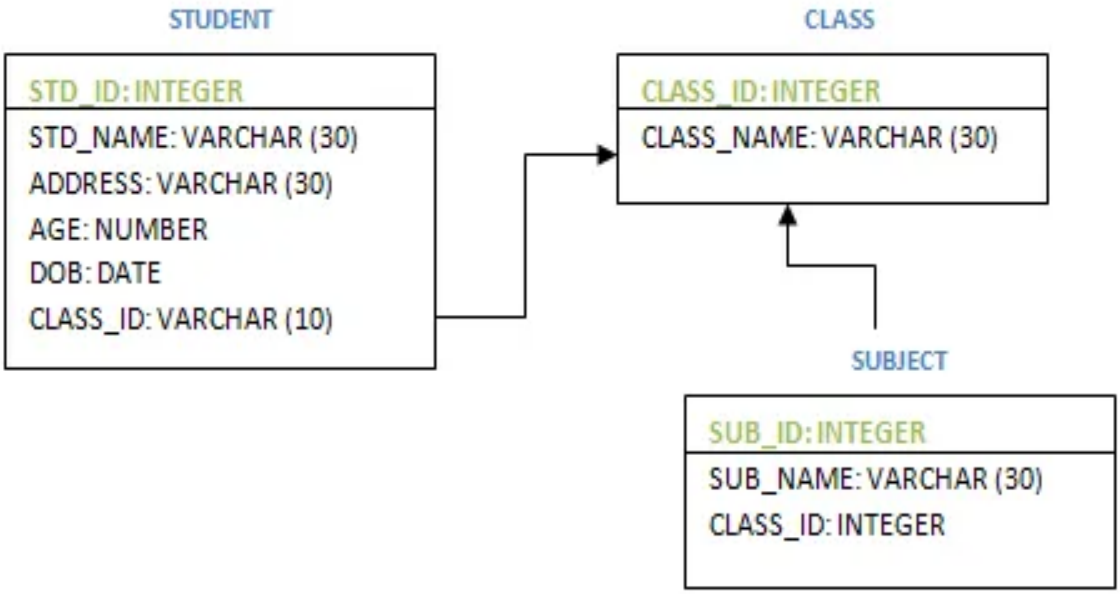

Рис. 1.4 – Фізична модель[5]

1.3 Типи БД. Реляційна модель

Існує декілька моделей або типів баз даних, основні з них це: мережна, ієрархічна, об’єктно-орієнтована та найбільш поширена у використанні реляційна.[10]-[15] Розглянемо їх детальніше.

Мережева або графічна називається БД, де кілька файлів або записів учасників можуть бути пов’язані з кількома файлами власників та навпаки.

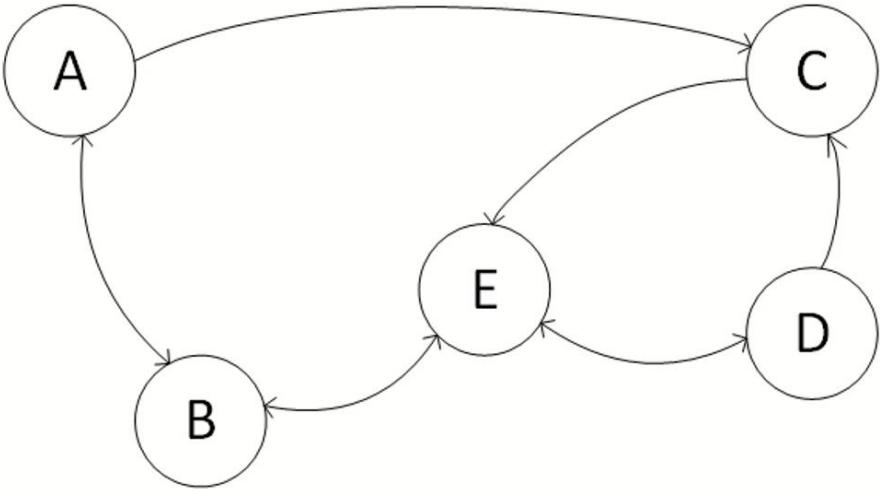

Рис. 1.5 – Схематичне зображення мережевого типу

Ця схема, наприклад, можна використати для візуалізації дороги між пунктами населення. Але область використання у порівнянні з іншими типами БД крайнє малий. Приклади програм де використовується: AllegroGraph, Amazon Neptune, JanusGraph, тощо.

Ієрархічна використовує відношення «один-до-багатьох» для елементів даних, тобто деревовидну структуру, яка пов’язує декілька елементів з одним первинним записом «власник» («батька»). Таку БД, на приклад, можна побачити в різних файлових системах, реєстр WINDOWS, тощо. Приклади програм де використовується: Apache Directory, OpenLDAP, BaseX і т.д.

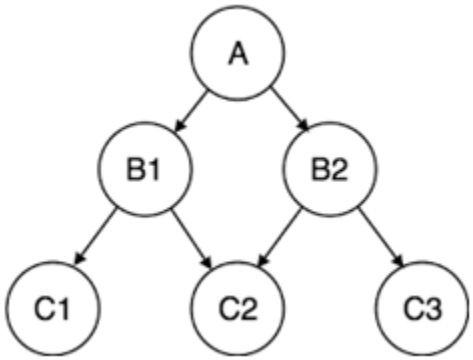

Рис. 1.6 – Схематичне зображення ієрархічного типу[11]

Об’єктно-орієнтована база даних, маніпулює інформацією, представленою об’єктами. Також слід відмітить, що деякі реляційні бази даних підтримують об'єктно-орієнтований підхід (наприклад, Oracle Database). Програмне забезпечення: InterSystems Cache, Google Cloud Storage для Firebase, dBASE PLUS, Apache OODT, та ішні.

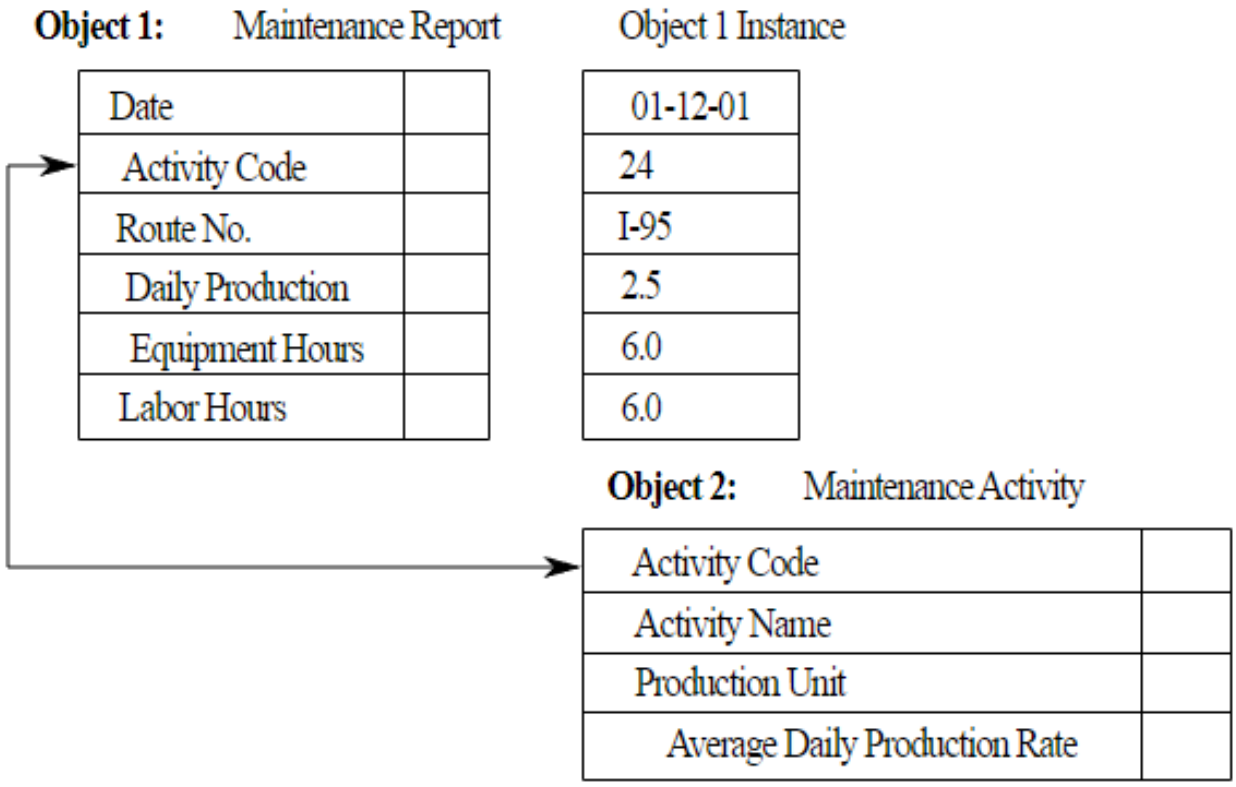

Рис. 1.7 – Схематичне зображення об’єктно-орієнтованого типу[12]

Найбільш поширені моделі являються саме реляційні, в силу того що є універсальною для багатьох задач в основному використовуються в комерційних програмах обробки даних . У перше була описана Едгаром Ф. Коддом у 1969 році. [11-15]Реляційна конструкція бази даних відповідає властивостям ACID (атомність, послідовність, цілісність і довговічність), необхідні для проектування бази даних. Передбачає використання сервера у програмах для вирішення проблем управління даними.

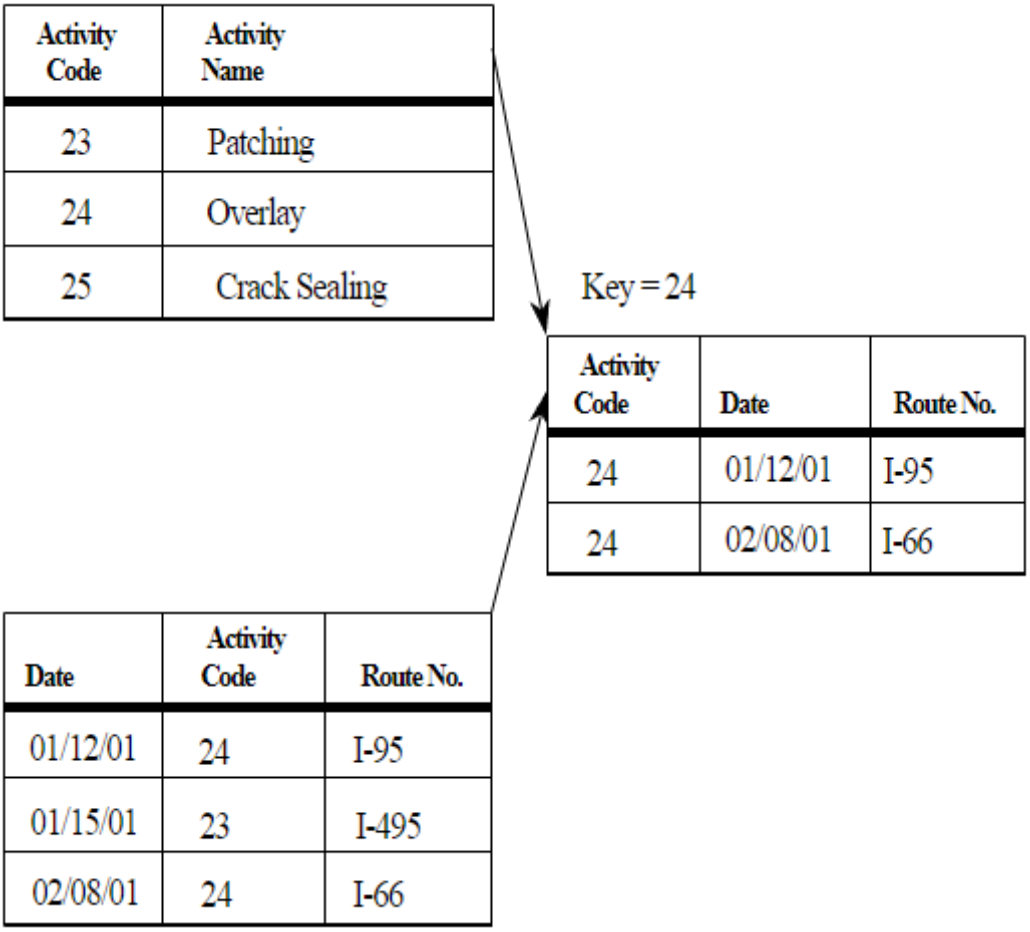

Рис. 1.8 – Схематичне зображення реляційного типу [11]

Цікаво, що ця конструкція одночасно описує 3 аспекти: структуру, правила цілісності та правила маніпулюванням. Головними фактами являються те, що дана модель є логічною, а не фізичною(1), тобто описується структури за допомогою спеціальних схем, не потрібно описувати конкретні дії СУБД по керуванню якимись файлами, записами всередині і т.д. що значно полегшило та прискорило розробку та підтримує як декларативний(це класична мова SQL в якому вказуємо СУБД лише кінцевий результат), так і імперативний підходи(Більш схожий на звичайну мову програмування, не сумісний між різними системами управління)(2).

```
CREATE TRIGGER `upd_date_ai` AFTER INSERT ON `news`
 FOR EACH ROW BEGIN
  DECLARE old_last_date int;
  SET old_last_date = (SELECT `nr_last_date`
   FROM `news_rubrics` WHERE
`nr_uid`=NEW.`n_parent`);
   IF old_last_date < NEW.`n_dt`
    THEN
     UPDATE `news_rubrics`
     SET `nr_last_date` = NEW.`n_dt`
      WHERE `nr_uid`=NEW.`n_parent`;
   END IF;
 END
```

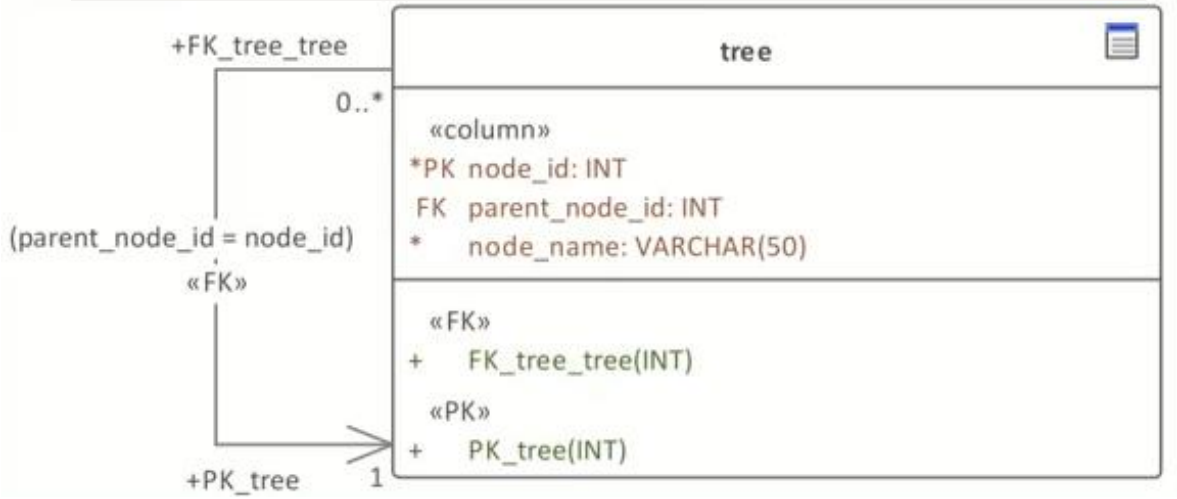

Рис. 1.9 – (1) Логічна модель, (2) Імперативний підхід.[16]

Про переваги та недоліки наведено в таблиці 1.

Таблиця 1. – Переваги та недоліки реляційної БД.

Переваги	Недоліки
Базується на простому наборі базових структур.	Реляційні бази даних вимагають багато пам’яті та потужності центрального процесора.
Використовує суворі математичні підходи.	Важко мати справу у разі великих баз даних.

Передбачає незалежність від внутрішніх структур.	Деякі структури (дерева, графіки тощо) важко реалізувати.

Але не дивлячись на такі недоліки реляційна модель існує вже понад 50 років і залишається один із найголовнішим з підходів по керуванню даними та БД і як показує практика буде залишатися такою в найближчому майбутньому.

Одне з фундаментальних визначень в реляційній БД – зв’язки, яке представляє собою об’єднання між суб’єктами.

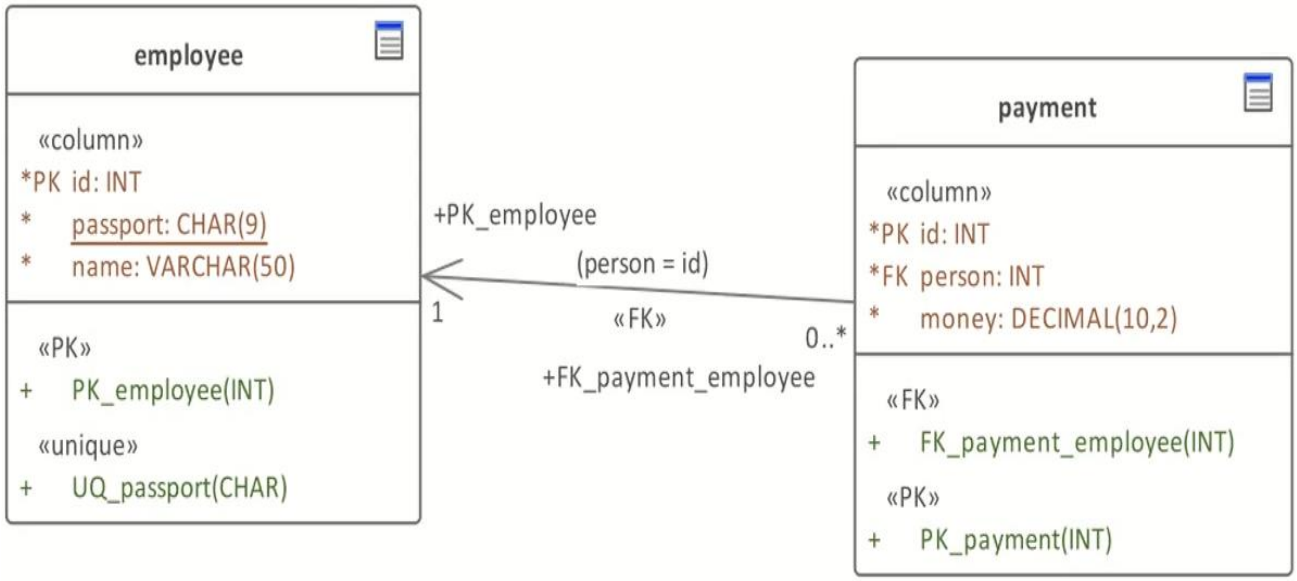

Рис. 1.10 – Приклад зв’язка між двома суб’єктами.[16]

Технічно, ці відносини засновані на міграції ключів. На рис. 1.11 наведено як саме первинний ключ (Primary Key) з «батьківського відношення» (Parent Relation) мігрує до «дочірнього» (Child Relation) який становиться зовнішнім (Foreign Key).

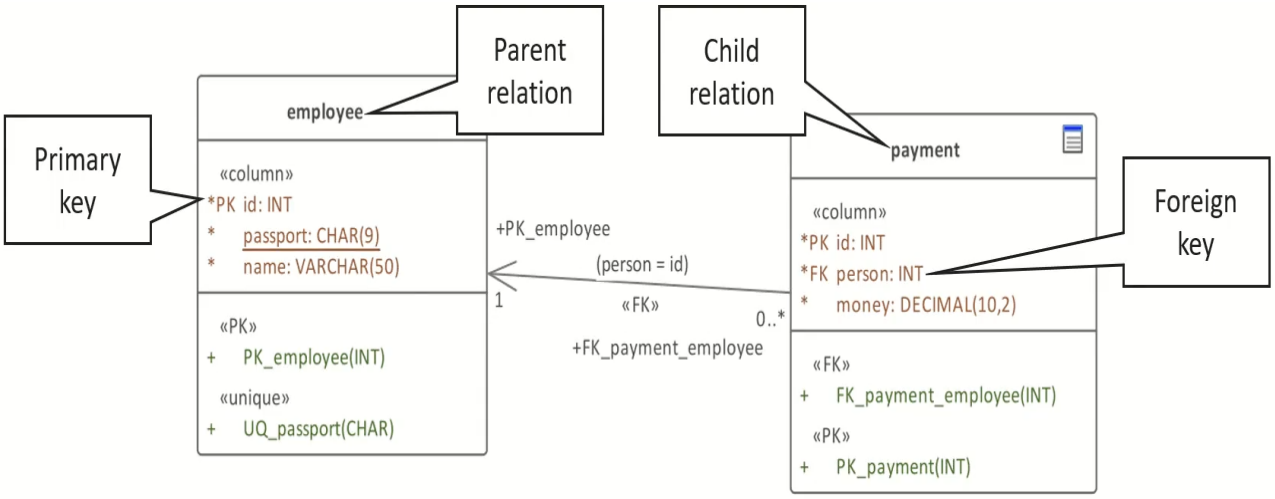

Рис.1.11 – Приклад міграції ключа.[16]

Приклади типів зв'язку :

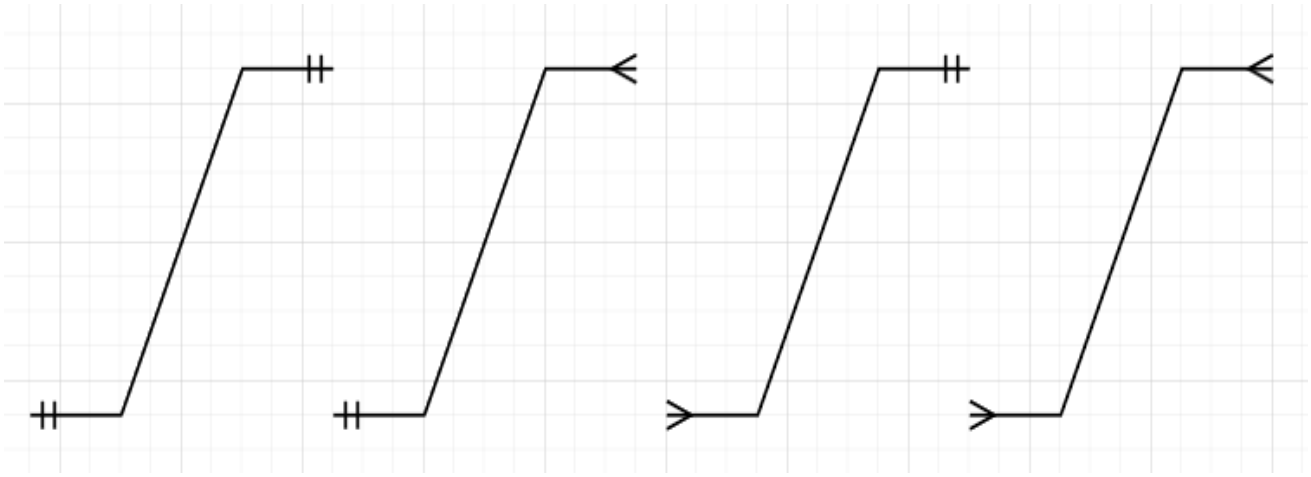

Рис. 1.12 один – до – одного, один – до – багатьох, багато – до – одного, багато – до – багатьох.

Нижче (Рис. 1.13) наведено базу даних з декотрими зв'язками, 1 працівник (“Employee”) може мати декілка платіжок (“Payment”) тоді ми отримаємо один – до – багатьох (“1 – to - M”) але якщо читати навпаки тобто багато платіжок належить 1 працівнику – багато-до-одного (“M – to - 1”).

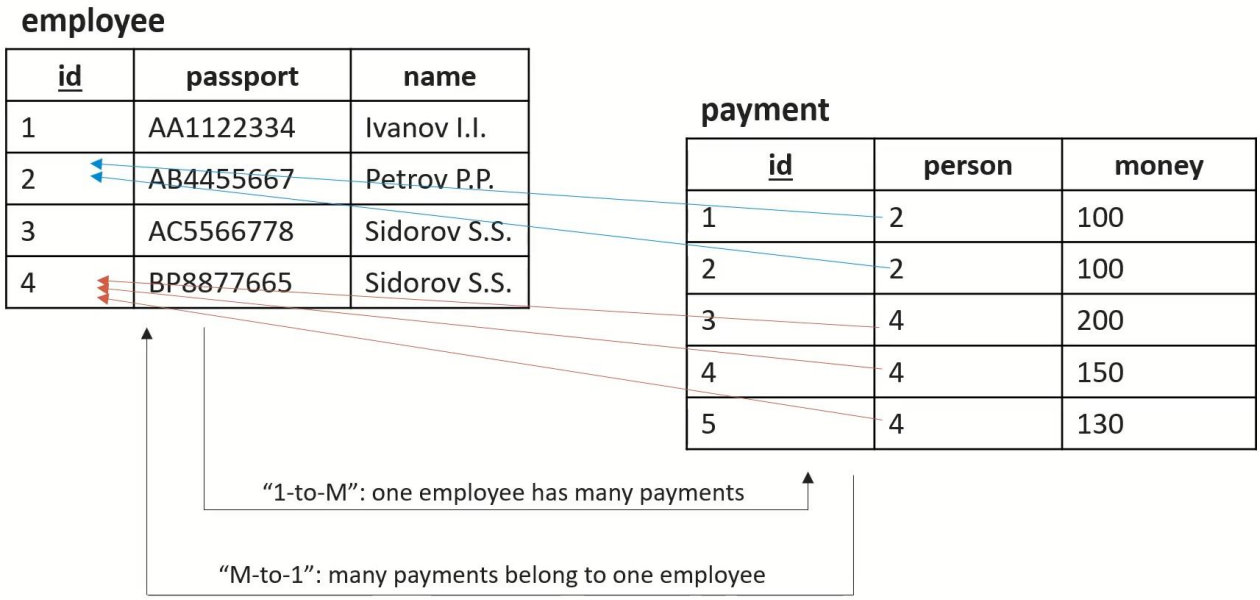

Рис. 1.13 – Приклад зв’язків.[16]

Кардинальність відноситься до максимальної кількості посилань, які можуть бути встановлені між екземплярами різних сутностей. Порядковий номер, у свою чергу, є мінімальною кількістю зв'язків між екземплярами двох сутностей.

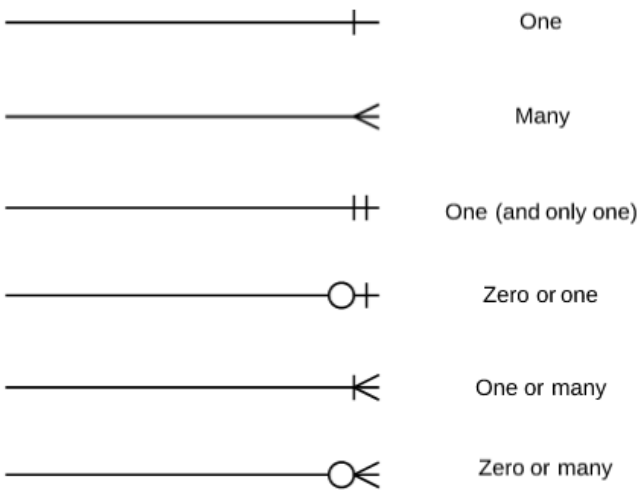

Рис. 1.14 – Види кардинальності та ординальності (1. – один,2. – багато,3. – один і тільки один,4. – нуль або один,5. – один або багато,6.- нуль або багато).

1.4 Нормалізація даних

Нормалізація БД сприяє з’єднанню пов'язаних даних в одну єдину таблицю. [15] Абиякі посередковані атрибутивні або пов’язані дані знаходяться в різноманітних таблицях і пов'язані логічними зв'язками.

Нормальні форми:

1. Перша форма (яка є головною структурованої системою БД): всяка таблиця мусить мати власний головний ключ; уникати дублювання груп; атомарність, тобто всякий атрибут мусить мати лише одне поняття замість набору.

2. Друга форма: проекція БД мусе відповідати усім дотриманням 1 форми; дані, які знову з'являються в певних рядках, видаляються з таблиці, має первинний ключ, усі атрибути залежать від первинного ключа націло а не від якійсь її частині.

3. Третя форма (потребує, щоб показники в сутностях підлягали тільки від основного ключа): Проекція БД мусить відписувати всім критерієм 1 нормальної форми.

4. Форма Бойса-Кодда: таблиця в 3 НФ, ключові не повинні залежати від не-ключових.

5. Четверта форма: у схемі бази даних не має бути нетривіальних багатозначних залежностей наборів атрибутів, за винятком наднабору ключів-кандидатів.

6. П'ята форма вимагає, щоб асоціація не мала нетривіальних залежностей, які не підкоряються обмеженням ключів. В 5-нормальній формі розглядається тільки в тому випадку, якщо вона знаходиться в 4NF і кожна асоціативна залежність викликана її ключами-кандидатами.

РОЗДІЛ 2. ПОБУДОВА БАЗИ ДАНИХ

“ЦЕНТР ЗОРУ” – сучасна приватна офтальмологічна клініка, яка надає платні послуги з діагностики, корекції та лікування зору. Задача створити реляційну БД для збирання статистичної інформації, виводу певних звітів.

Для побудови спочатку треба розподілити елементи інформації на основні сутності, оберу атрибути (перелік назв таблиць): послуги клініки - 'Service' , пацієнти - 'Patient', записи візитів пацієнтів до лікарів - 'Visit', лікарі - 'Doctor', його кваліфікація 'Category', графік роботи 'Schedule' також зв'язок між послугами, що надаються клінікою, і кваліфікацією лікаря 'm_2_m_service_category', згідно третьої нормальної форми можна створити саме такий набір таблиць.

2.1. Первинні та зовнішні ключі

Стовпець повинен мати свій унікальний ідентифікатор – первинний ключ (PK – Primary Key) по якому буде можна зсилатись на данні конкретної таблиці. Треба зауважити, що у одній таблиці можливо наявність лише одного первинного ключа та усі стовпці з обмеженням PK повинні мати ознаку NOT NULL. Створимо PK для таблиць:

- 'Service' – 's_ID';
- 'Patient' – 'p_ID';
- 'Visit' – 'v_ID;
- 'Doctor' – 'd_ID';
- 'Category' – 'c_ID';
- 'Schedule' – 'sc_ID';

Крім первинних ключів-ідентифікаторів існують зовнішні (FK - Foreign Key). Вони дозволяють встановити зв'язок між таблицями. Зовнішній ключ встановлюється зазвичай для стовпців із залежної, підлеглої таблиці, та вказує на одну із стовпців із головної. Зв’язую таблиці за допомогою зовнішніх ключів:

- В першу чергу щоб зробити новий запит, в таблиці 'Visit' зроблено ключ 'v_p_ID' який зв’язаний з основним ключем з таблиці 'Patient' – 'p_ID' і надає особову інформацію пацієнта.
- Щоб зв’язати з 'Doctor' таблицею, створимо зовнішній ключ 'Visit' - 'v_d_ID' який вказує на 'd_ID', яка дасть нам інформацію про лікаря, який буде обслуговувати конкретного пацієнта.
- Щоб дізнатись послугу яка надається з 'Service' , створено в 'Visit' – 'v_s_ID' що зв’язує з основним ключем-ідентифікатором послуги 's_ID'.
- Кожний доктор має свою певну конкретну кваліфікацію тобто категорію , від цього залежить його ряд послуг який він може надати. Зв’язую 'Doctor' FK – 'd_c_ID' з 'Category' PK - 'c_ID'.
- Окрім цього лікарі мають різний графік роботи він внесений в FK - 'd_sc_ID' який представить цю інформацію з таблиці 'Schedule' PK – 'sc_ID'.
- Таблиця 'm_2_m_service_category' не має своєї індивідуальної інформації вона лише допоміжний зв’язок між таблицями FK – 's_m_ID' до 'Service' – PK 's_ID' і FK – 'c_m_ID' представляє дані з 'Category' – PK 'c_ID'. Такий спосіб зв’язку називають багато до багатьох (many-to-many). 'm_2_m_service_category' має багато послуг та кваліфікація докторів які можуть надати цю послугу, ця виносна таблиця зроблена аби не дублювати данні, наприклад в таблиці 'Doctor' якщо мала би ще допоміжну комірку з послугами, яку лікар вміє надавати (напряму зв’язана з 'Service') то таблиця мала велике розширення. За нормальними формами, ми не можемо робити перелік в рамках однієї комірки, наприклад доктор з вищою категорією котрий вміє робити багато ряд послуг, дану інформацію прийшлось додавати що не комфортно для сприйняття та використання такими даними.

2.2. Побудова ER-Діаграми

Перейдемо до таблиць і докладніше представлю які комірки буде вміщати кожна таблиця.

1. 'Patient': ключ ідентифікатор – 'p_ID' , ім’я – 'p_name', прізвище – 'p_surname' та по батькові – 'p_middle_name' клієнта. Також стать людини – 'p_sex', його дату народження – 'p_birth_date' та його повний вік – 'p_age', також номер телефону – 'p_phone_number' для зв’язку с пацієнтом.

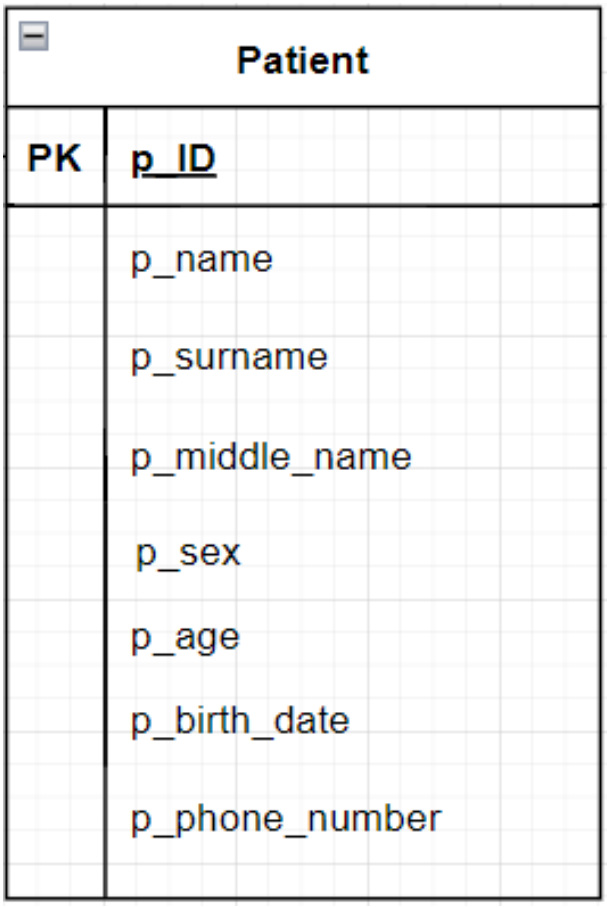

Рис. 2.1 - вигляд таблиці 'Patient'.

2. 'Doctor': ключ-ідентифікатор – 'd_ID', ПІБ лікаря – 'd_name','d_surname' ,'d_middle_name', кваліфікацію – 'd_s_ID', графік роботи – 'd_sc_ID', також досвід роботи (скільки років працює в своїй галузі) – 'd_experience', його номер кабінету – 'd_room', електронна пошта – 'd_e_mail та номер телефону лікаря – 'd_phone_number'.

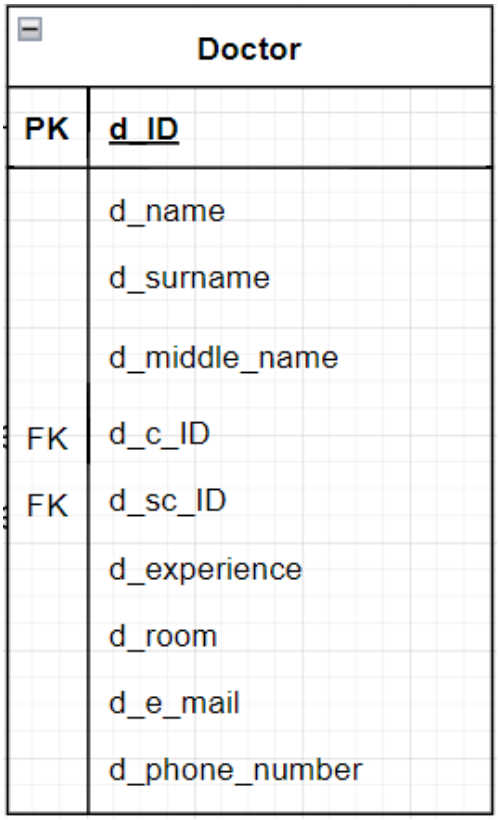

Рис. 2.2 – вигляд таблиці 'Doctor'.

3. 'Schedule': вказує який робочий графік повний, перша половина дня чи друга – 'sc_ID', початок робочого часу – 'sc_start' та кінець – 'sc_end'.

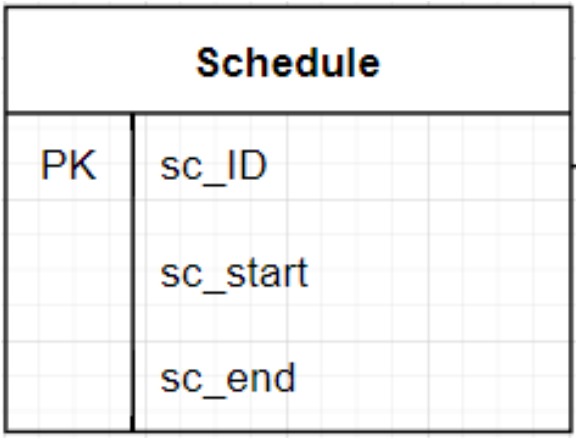

Рис. 2.3 - вигляд таблиці 'Schedule'.

4. 'Visit': номер візиту пацієнта – 'v_ID', дата візиту клієнта – 'v_date', ідентифікатор пацієнта – 'v_p_ID', лікар який обслуговує – 'v_d_ID',ключ сервісу – 'v_s_ID', діагноз пацієнта – 'v_diagnosis'.

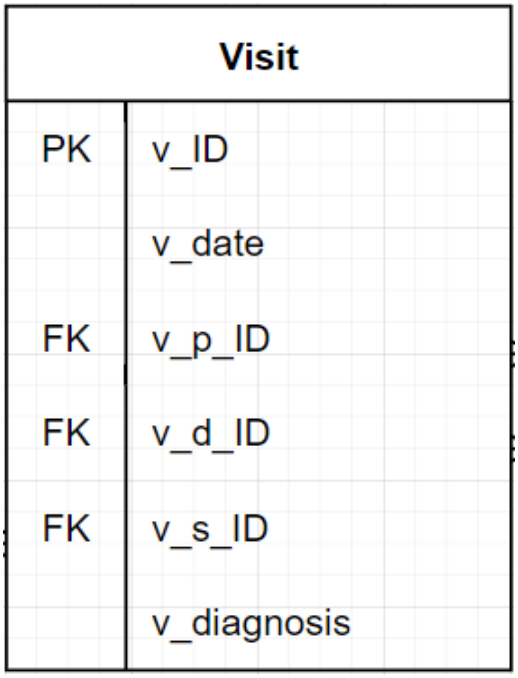

Рис. 2.4 - вигляд таблиці 'Visit'.

5. 'Category': ключ-ідентифікатор – 'c_ID' і інформація про кваліфікацію лікаря – 'c_description'.

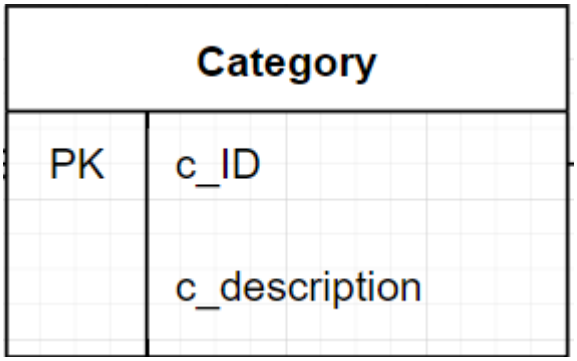

Рис. 2.5 - вигляд таблиці 'Category'.

6. 'Service': ключ-ідентифікатор – 's_ID', послуга офтальмологічної клініки – 's_description', ціна за конкретну послугу яка надається – 's_price'.

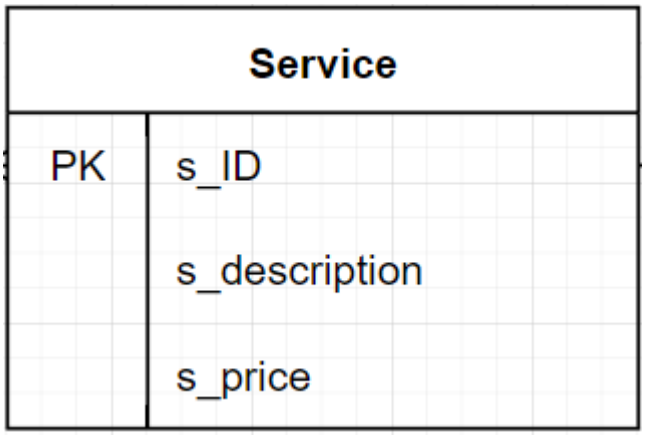

Рис. 2.6 - вигляд таблиці 'Service'.

7. 'm_2_m_service_category': зв’язок послуги клініки 's_m_ID' і категорії доктора 'c_m_ID'.

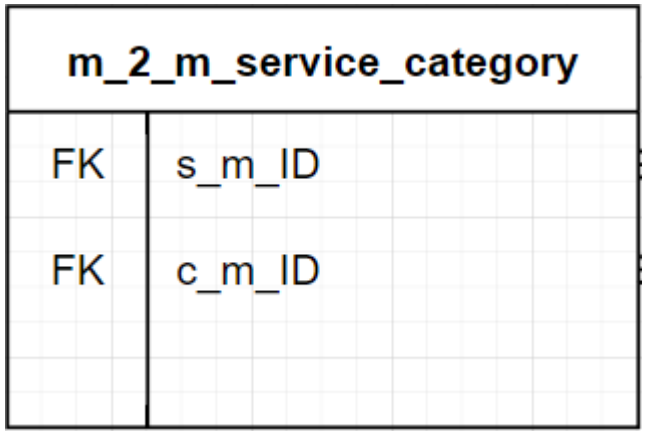

Рис. 2.7 - вигляд таблиці 'm_2_m_service_category'.

Тепер потрібно візуально зв’язати ці таблиці між собою. Для цього мною було використано побудова схем онлайн Flowchart Maker & Online Diagram Software [20].

Один пацієнт може відвідати поліклініку декілька разів тому використаємо зв’язок один – до – багатьох або нуль від основного ключа 'Patient' до зовнішнього 'Visit' – 'v_p_ID'(Рис 2.8).

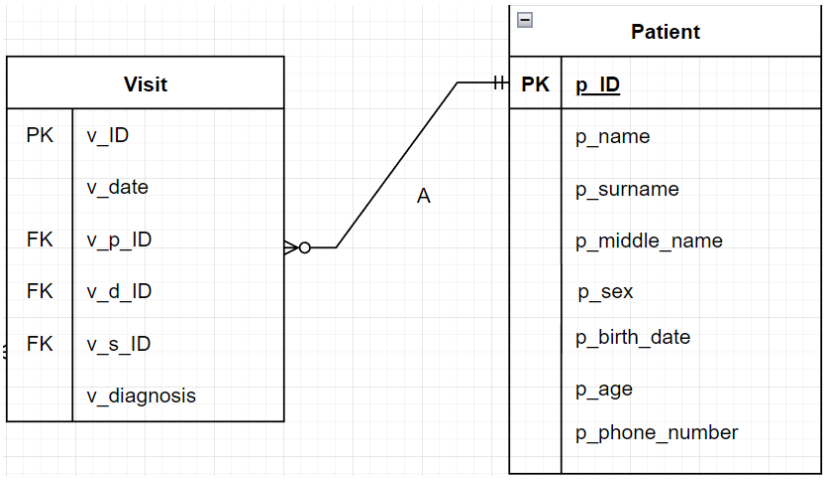

Рис. 2.8 - зв'язок один – до – багатьох або нуль між 'Patient' та 'Visit'.

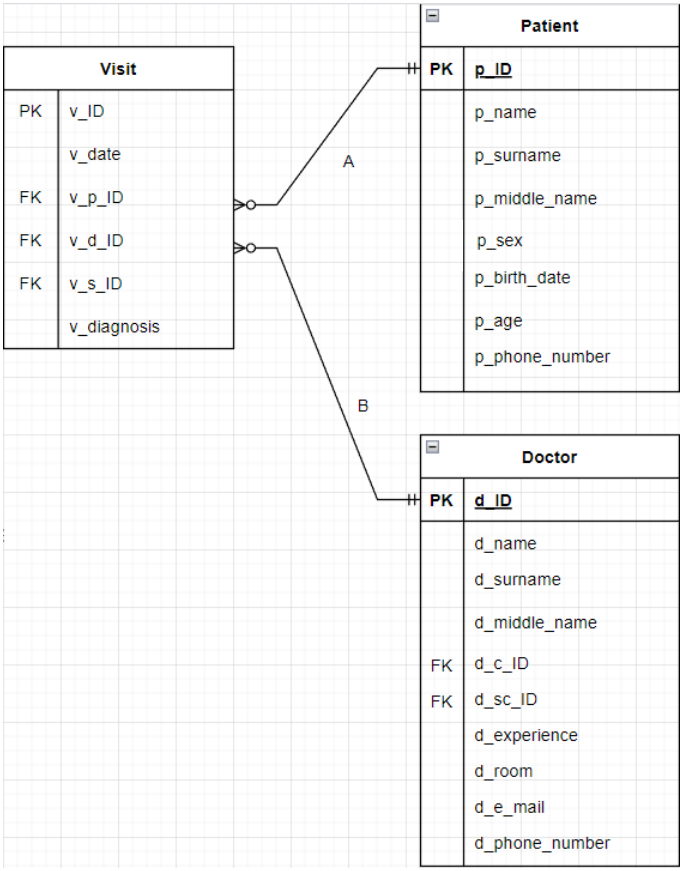

Рис. 2.9 - зв'язок один – до – багатьох або нуль між 'Doctor' та 'Visit'.

Один лікар може мати декілька пацієнтів в день підходить зв’язок один – до – багатьох або нуль від основного ключа 'Doctor' до зовнішнього 'Visit' – 'v_d_ID' (рис 2.9).

Декілька офтальмологів можуть мати одну і ту саму кваліфікацію, тож приєдную за допомогою один – до – багатьох або нуль від основного ключа 'Category' – 'c_ID' до зовнішнього 'Doctor' – 'd_c_ID'. (Рис 2.10).

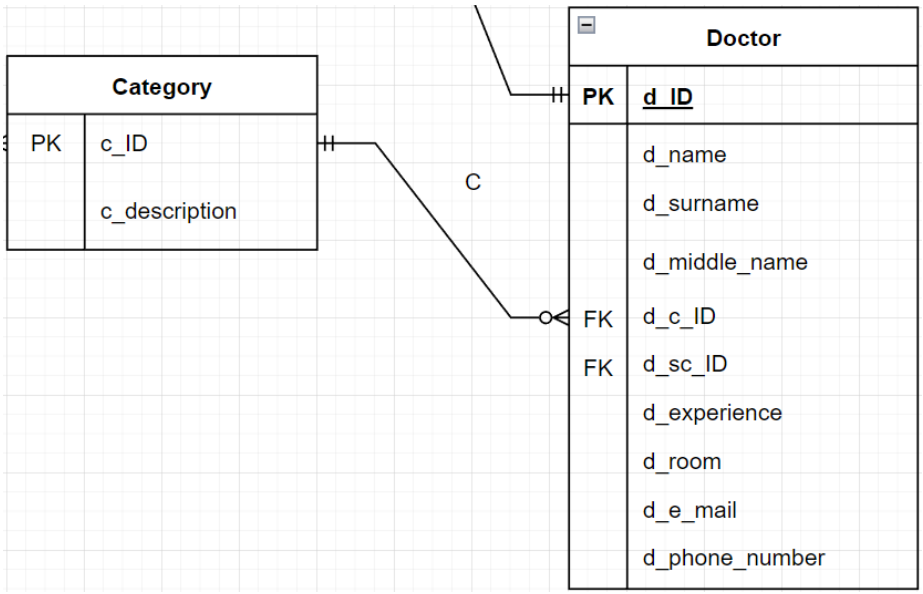

Рис. 2.10 - зв'язок один – до – багатьох або нуль між 'Category' та 'Doctor'.

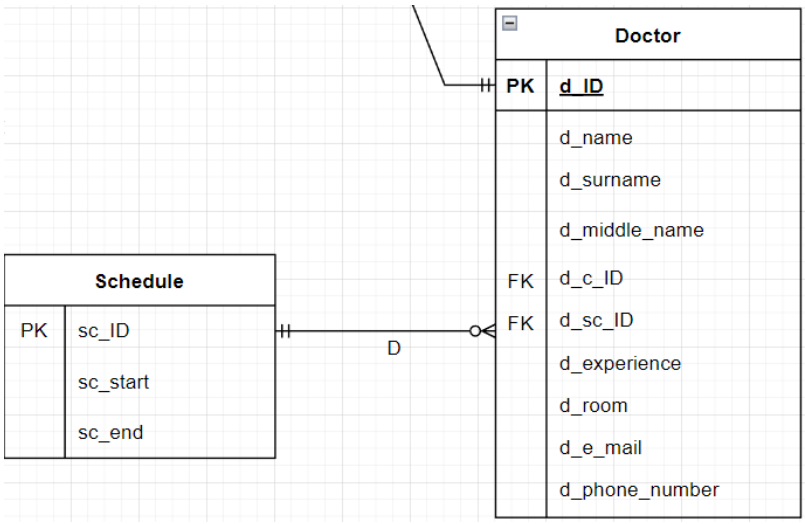

Рис. 2.11 - зв'язок один – до – багатьох або нуль між 'Schedule' та 'Doctor'

В сутності 'Schedule' знаходяться певний графік по якому можуть працювати кілька доктора з цього провожу зв’язок з графіку 'sc_ID' до 'Doctor' – 'd_sc_ID' один – до – багатьох або нуль (Рис 2.11).

Як же було пояснено вище мною сутність 'm_2_m_service_category' була створена для приєднання сервісу та категорії (багато послуг може предоставити багато кваліфікацій). Для цього використаємо 2 зв’язка один – до – багатьох або нуль (Рис 2.12).

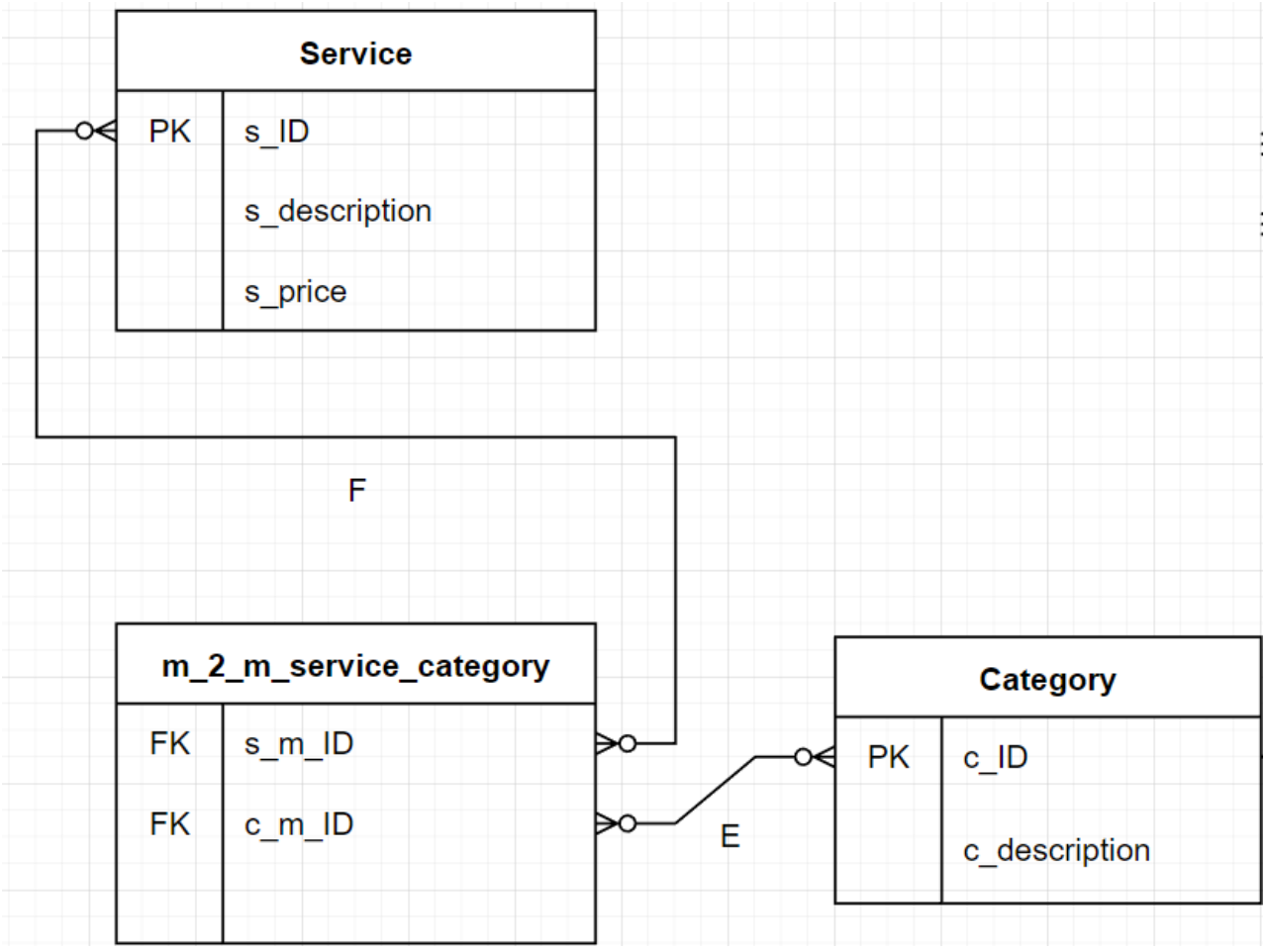

Рис. 2.12- зв'язки багато – до – багатьох або нуль між 'Category' та 'Service' через допоміжну таблицю 'm_2_m_service_category'.

Останній зв’язок вказує на те що одна послуга може бути обслугована для багатьох пацієнтів (Рис 2.13) .

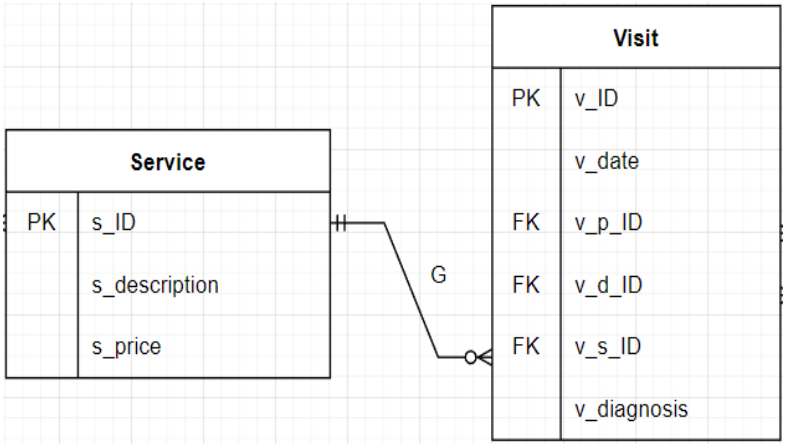

Рис. 2.13 - зв'язок один – до – багатьох або нуль між 'Service' та 'Visit'.

Готова ER – діаграма для офтальмологічної клініки “ЦЕНТР ЗОРУ” продемонстровано нижче:

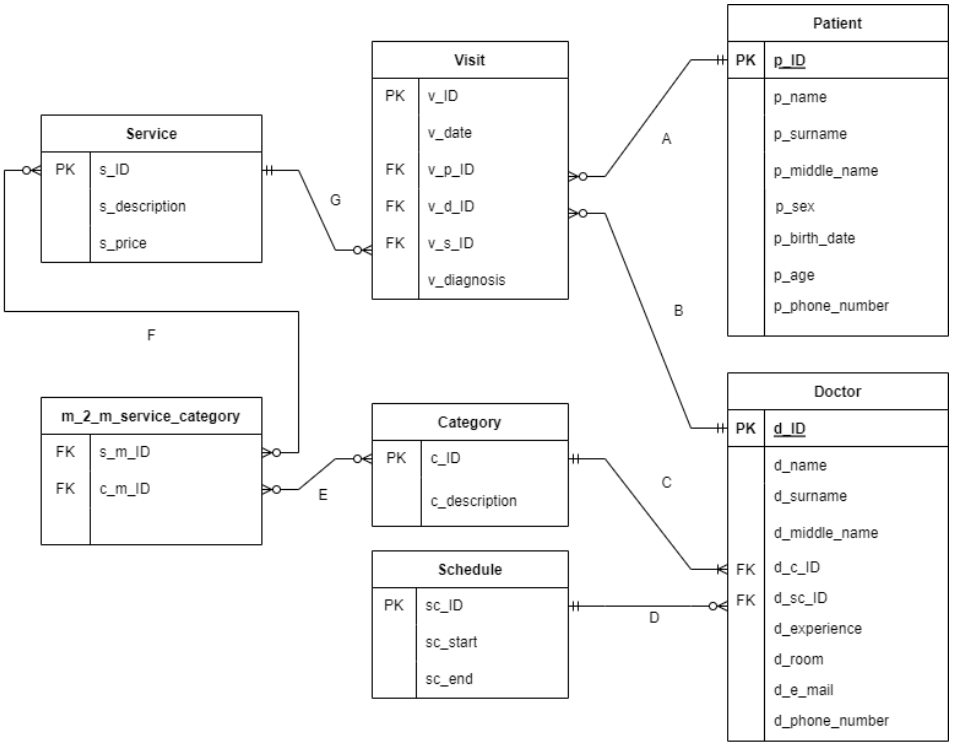

Рис. 2.14 - ER-діаграма для офтальмологічної клініки.

2.3 Тригери

Процедура, яка виконується з настанням певної події в реляційної бази даних, що відкомпілювалася називають тригером. Це інструмент який використовують для підтримання цілісності даних в БД, збережна процедура котра автоматично запускається із сервером при намаганні змінення даних в таблицях, з котрими тригери зв’язані.

За допомогою даного інструмента можна перевірити коректність даних які введені, нагадати про важливість виконання декотрих дій при змін в таблиці також підтримує реплікації.

Недоліками тригерів являються: їх складність реалізувати та проектування; при перенесенні елементи функцій у БД в подобі декількох тригерів може призводити до маскування декотрих можливостей при цьому користувачі не зможуть маніпулювати всіма процесами; при виконання команд по зміні інформації в таблицях, СУБД звіряє тригерну умову для рішення необхідності її запуску конкретно цієї команди, здійснення таких операцій проявляється на ефективності СУБД а при максимальної навантаженні зниження продуктивності може стати дуже помітною.

Наявні два параметри тригерів, виконання після вдалого реалізування команди(AFTER), викликання замість команди(INSTEAD).

Існують три типи тригерів:

1. INSERT TRIGGER – запускається під час спроби вставити дані за допомогою команди INSERT.
2. UPDATE TRIGGER - запускається під час спроби змінити дані за допомогою UPDATE.
3. DELETE TRIGGER - запускається при спробі видалити дані за допомогою DELETE.

Треба зауважити що всередині тригера не дозволяється, наприклад, створювати, змінювати і видаляти БД також відновлювати резервні копії БД. Здійснення даних команд не припускається, так як не може бути відмінено у разі відкату транзакції, в якому діє тригер. У разі непотрібності тригера існує команда його видалення DROP TRIGGER {ім'я_тригера} [,..n].

2.4 Реляційна БД в СУБД

Реляційна база даних була створена через додаток DB Browser:

Для коректного введення інформації в БД, мною було створено деякі тригери.

```
CREATE TRIGGER wrong_phone_number
    BEFORE INSERT ON Patient
BEGIN
    SELECT
        CASE
    WHEN NEW.p_phone_number NOT LIKE '380(__)___-__-__' THEN
        RAISE (ABORT, 'Невірний формат номера телефону' )
        END;
        END;
```

Рис. 2.15 - тригер для заповнення номера телефону.

На даному рис. 2.15, відображено код створення тригеру у якому в разі невірного заповнення номера телефону пацієнта буде з’являтися помилка 'Невірний формат номера телефону'. Продемонструю на рис. 2.16:

SQL 1

```
INSERT INTO Patient (p_ID,p_name,p_surname,p_middle_name,p_phone_number)
VALUES('101','Степан','Роєнко','Тимофійович','380(435)513-12-145')
```

```
-- ВЫПОЛНЕНИЕ ВСЕ В 'SQL 1'
--
-- At line 1:
INSERT INTO Patient (p_ID,p_name,p_surname,p_middle_name,p_phone_number)
VALUES('101','Степан','Роєнко','Тимофійович','380(435)513-12-145')
-- Result: Невірний формат номера телефону
```

```
INSERT INTO Patient (p_ID,p_name,p_surname,p_middle_name,p_sex,p_birth_date,p_age,p_phone_number)
VALUES('101','Степан','Роєнко','Тимофійович','Ч','04.07.2004','18','380(55)513-12-14')
```

```
-- Result: 5 values for 8 columns
-- ВЫПОЛНЕНИЕ ВСЕ В 'SQL 1'
--
-- At line 1:
INSERT INTO Patient (p_ID,p_name,p_surname,p_middle_name,p_sex,p_birth_dat
VALUES('101','Степан','Роєнко','Тимофійович','Ч','04.07.2004','18','380(55
-- Result: запрос успешно выполнен. Заняло 1мс, 1 строк изменено
```

Рис. 2.16 - приклад спрацювання тригера .

За схожим кодом, зроблено тригер з коміркою електронна пошта в таблиці лікарів.

```
CREATE TRIGGER wrong_e_mail
   BEFORE INSERT ON Doctor
BEGIN
   SELECT
      CASE
    WHEN NEW.d_e_address NOT LIKE '%_@__%.__%' THEN
      RAISE (ABORT,'Невірно набрана електронна пошта')
       END;
END;
```

Рис. 2.17 - тригер для заповнення електронної пошти.

```
RT INTO Doctor (d_ID,d_name,d_surname,d_middle_name,d_c_ID,d_sc_ID,d_experience,d_room,d_e_address,d_phor
ES('21','Виталий','Будко','Максимович','1','Повний день','7','120','forexample.com','380(55)513-21-41')
```

```
-- At line 1:
INSERT INTO Doctor (d_ID,d_name,d_surname,d_middle_name,d_c_ID,d_sc_ID,
VALUES('21','Виталий','Будко','Максимович','1','Повний день','7','120',
-- Result: Невірно набрана електронна пошта
```

```
d_name,d_surname,d_middle_name,d_c_ID,d_sc_ID,d_experience,d_room,d_e_address,d_phone_number)
дко','Максимович','1','Повний день','7','120','forexample@mail.com','380(55)513-21-41')
```

```
INSERT INTO Doctor (d_ID,d_name,d_surname,d_middle_name,d_c_ID,d_sc_ID
VALUES('21','Виталий','Будко','Максимович','1','Повний день','7','120'
-- Result: запрос успешно выполнен. Заняло 0мс, 1 строк изменено
```

Рис. 2.18 - приклад спрацювання тригера.

Мною зроблено ще тригер (Рис. 2.19)котрий перевіряє правильність введення дати народження пацієнта, щоб уникнути помилки які можуть з’явитися при відсортуванні клієнтів через дану комірку.

```
CREATE TRIGGER wrong_birth_date
   BEFORE INSERT ON Patient
BEGIN
   SELECT
      CASE
    WHEN NEW.p_birth_date NOT LIKE '__.__.____' THEN
      RAISE (ABORT,'Невірно введено дату народження')
       END;
END;
```

Рис. 2.19 - тригер для дати народження.

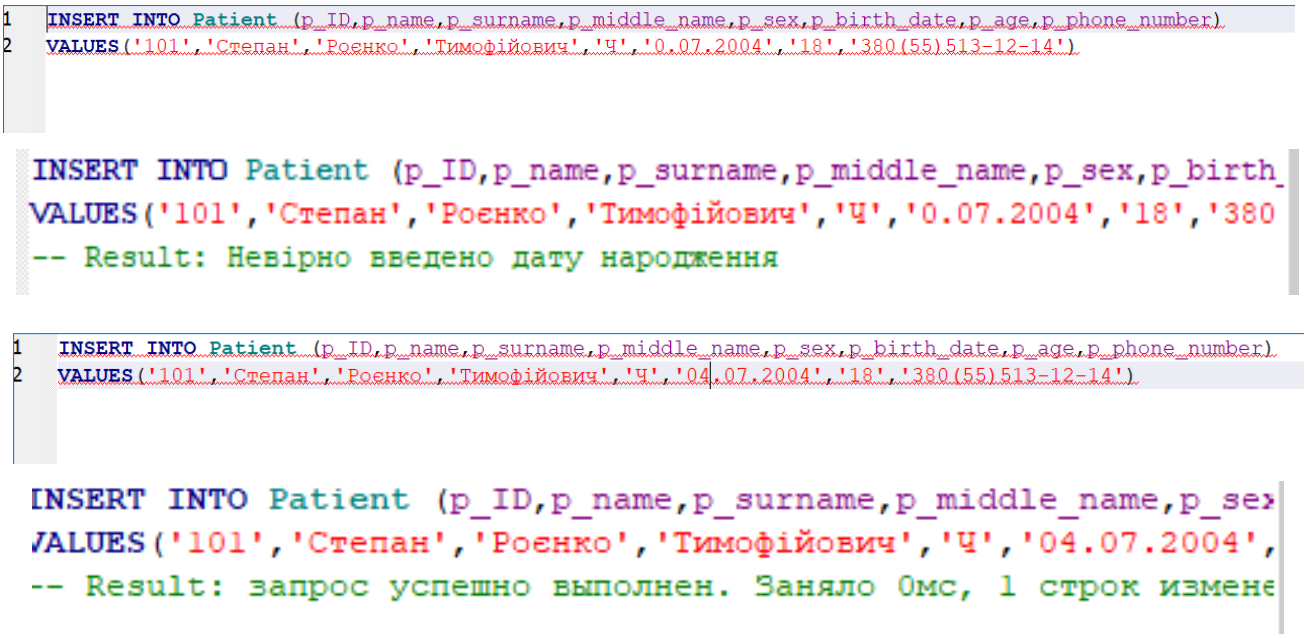

Рис. 2.20 - приклад спрацювання тригера.

Для заповнення ПІБ, сутностей 'Patient' і 'Doctor', були використані онлайн генератори [17][19].

На рисунку 2.21 дізнаємось як саме цей сайт працює: в ньому можна обрати стать людини, мову походження імен та прізвищ, за бажанням обрати першу букву імен та кількість яка потребується, як приклад створено навмання десять ПІБ людей чоловічої статі.

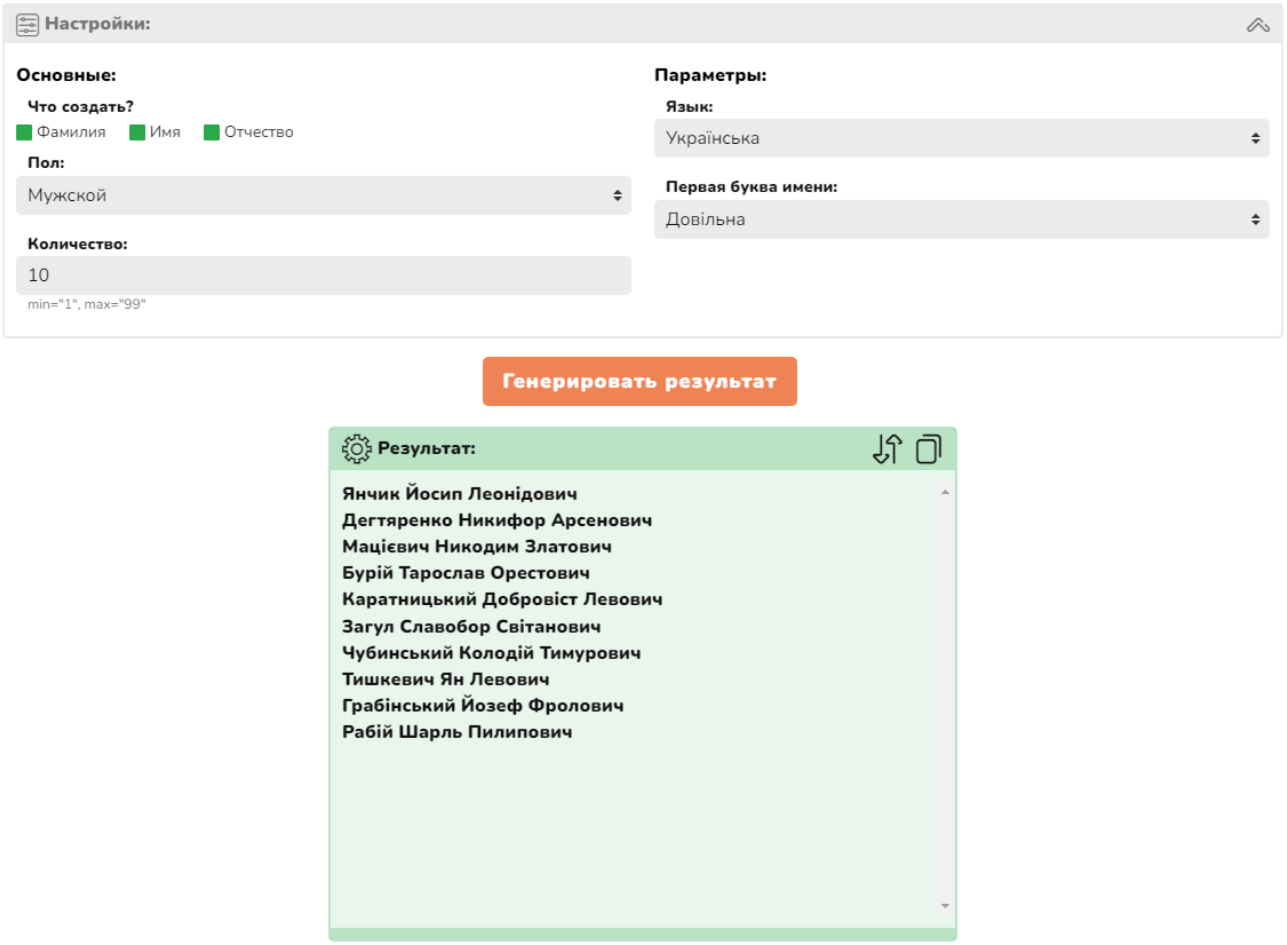

Рис. 2.21 – приклад створення клієнтів для БД.

Для внесення комірки 'p_birth_date' в таблиці 'Patient' використано генератор онлайн [18] для створення навмання дати народження: перша комірка обираємо кількість дат яка потрібна, далі обираємо інтервал між датами в яких буде генеруватися дати, за бажанням обрати день неділі, формат вигляду і локація для того щоб представити іншомовною мовою.

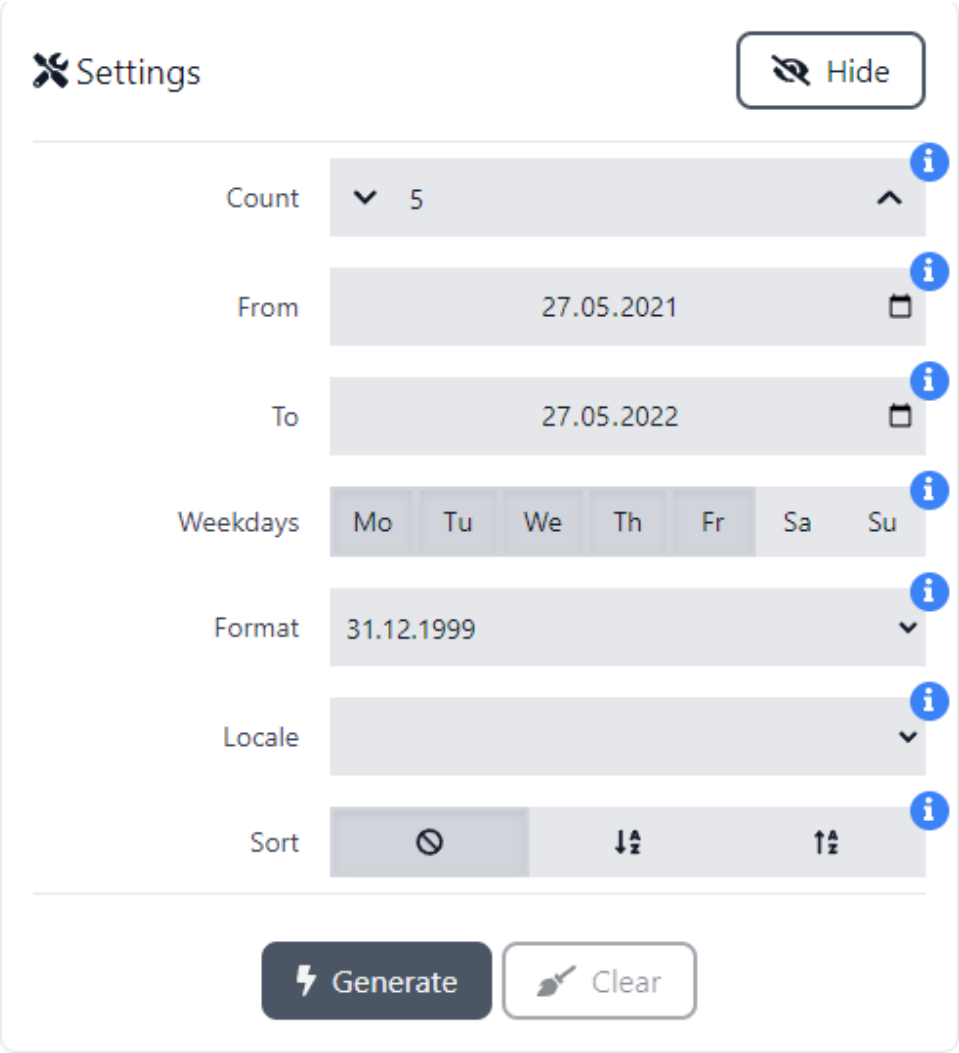

Рис. 2.22 – приклад генерації дат народження.

Також за допомогою іншого генератора онлайн [19] створено номери телефону для пацієнтів та лікарів. У ньому підбираємо код країни та міста, цифр в номері, формат телефону та кількість номерів, приклад наведено на Рис 2.23.

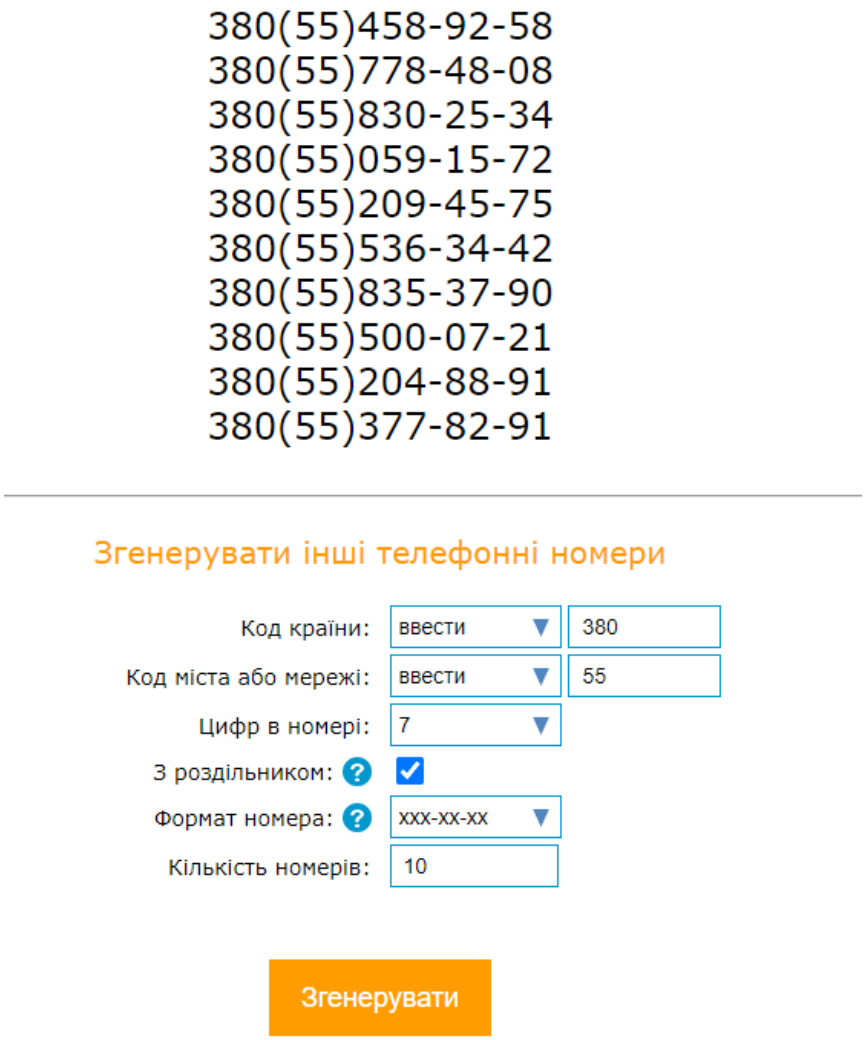

Рис. 2.23 – приклад генерації номерів телефонів.

Нижче представлено, кінцевий вигляд кожної заповненої комірки в кожній сутності реляційної БД в СУБД для клініки:

c_ID	c_description
Фи...	Фильтр
1	Хірург-офтальмолог
2	Дитячий офтальмолог
3	Лікар-офтальмолог вищої кваліфікаційної ...
4	Лікар-офтальмолог початкової кваліфікаційної ...

d_ID	d_name	l_surname	_middle_nam	d_c_ID	d_sc_ID	xperie	d_room	d_e_address	d_phone_number
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фи...	Фильтр	Фильтр	Фильтр
1	Нігослав	Кухар	Юхимович	1	Повний день	12	100	meiquacrallaubru-9859@...	380(99)413-32-74
2	Яснолик	Зубриц...	Любомиро...	2	Перша полови...	5	101	borecrapeimmi-2051@mai...	380(99)534-88-43
3	Йосеф	Кравчук	Володими...	2	Друга половин...	4	102	gehocuwoca-7624@mail.c...	380(99)184-66-64
4	Цвітан	Кривенко	Жданович	3	Повний день	6	103	zettucroizaque-8693@mail...	380(99)254-29-34
5	Макар	Нагірний	Олександр...	3	Друга половин...	6	104	nujatreubritto-4231@mail....	380(99)577-75-29
6	Стожар	Сеник	Максимович	1	Перша полови...	8	105	nuprennoimuyo-4717@ma...	380(99)549-49-28
7	Ян	Соколе...	Найденович	3	Повний день	11	106	wodemmolimo-4848@mail...	380(99)127-79-17
8	Шарль	Даниль...	Світанович	4	Перша полови...	9	107	hamottaxulu-6347@mail.c...	380(99)724-65-89
9	Йошка	Дмитріє...	Герасимов...	2	Повний день	7	108	brissapreubebe-9366@ma...	380(99)678-19-72
10	Щедрогост	Штокало	Полянович	2	Повний день	5	109	yexuhoippeila-7271@mail....	380(99)210-14-82

p_ID	p_name	p_surname	p_middle_name	p_sex	p_age	p_birth_date	p_phone_number
Фи...	Фильтр	Фильтр	Фильтр	Фил...	Фил...	Фильтр	Фильтр
1	Флор	Сахно	Вікторович	Ч	26	21.10.1996	380(55)908-46-41
2	Злат	Криворучко	Тихонович	Ч	25	09.09.1997	380(55)863-29-82
3	Остап	Кмета	Омелянович	Ч	44	03.05.1978	380(98)333-61-53
4	Івантослав	Салійчук	Чеславович	Ч	11	11.10.2011	380(95)205-20-22
5	Лука	Писаревський	Златович	Ч	33	25.04.1989	380(66)915-21-68
6	Вишеслав	Волянський	Златович	Ч	15	27.09.2007	380(99)766-93-95
7	Зіновій	Жоравки	Олександрович	Ч	22	16.09.2000	380(55)460-83-27
8	Йошка	Роменський	Тарасович	Ч	13	22.11.2009	380(99)347-97-92
9	Наслав	Гурик	Богданович	Ч	43	22.04.1979	380(66)442-86-66
10	Йонас	Лісовенко	Вікторович	Ч	52	01.05.1970	380(98)261-57-03

v_ID	v_date	v_p_ID	v_d_ID	v_s_ID	v_diagnosis
Фи...	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
1	07.11.2021	1	14	2	Вроджена катаракта
2	07.11.2021	5	6	6	Астигматизм
3	12.11.2021	34	18	1	Патології сітківки ока
4	12.11.2021	20	8	3	Далекозорість
5	12.11.2021	25	10	7	Далекозорість
6	12.11.2021	73	11	7	Катаракта
7	19.11.2021	80	19	5	Короткозорість
8	19.11.2021	98	5	7	Катаракта
9	20.11.2021	73	3	7	Короткозорість
10	10.12.2021	6	7	6	Катаракта
11	10.12.2021	44	18	1	Далекозорість
12	13.12.2021	26	9	7	Катаракта
13	13.12.2021	85	11	6	Глаукома

s_ID	s_description	s_price
Фи...	Фільтр	Фільтр
1	Діагностика зору	1500 грн
2	Хірургічне лікування катаракти	25000 грн
3	Хірургічне лікування глаукоми	20000 грн
4	Лазерне лікування глаукоми	14000 грн
5	Лазерна корекція	29000 грн
6	Естетична хірургія	10000 грн
7	Діагностика зору для дітей	1000 грн
8	Підбір окулярів	300 грн
9	Оптична когерентна томографія сітківки	800 грн
10	Установка контактних лінз	500 грн

sc_ID	sc_start	sc_end
Фильтр	Фильтр	Фильтр
Повний день	8:00	19:00
Перша половина дня	8:00	13:30
Друга половина дня	13:30	19:00

Рис. 2.24 – приклад реляційної БД в додатку DB Browser .

РОЗДІЛ 3. ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ПРИ РОБОТІ ЗА КОМП'ЮТЕРОМ

Працюючи за комп'ютером, рекомендовано дотримуватися правил:

- Відстань стільця до столу так щоб між краєм столу та Вами був вільний простір не менше 5 см, налаштування висоти стільця для комфорту під час роботи.

- При роботі з ПК потрібно держати спину рівно.

- Плечі потрібно розслабити та вільно опущені аби руки були розслабленні.

- Ноги повинні опиратися на підлогу або на спеціальну підставку під кутом 90 градусів ;стегна повинні розташовані під кутом 90 градусів до тулуба, гомілки під прямим кутом до стегон; використовуйте підлокітники лише під прямим кутом.

- Клавіатура перед роботою треба посмістити так щоб між вами була відстань приблизно 10 - 25 см, у конструкції клавіатури має передбачатися опорний пристрій, кут нахилу від 5 до 15 градусів.

- Прослідкуйте за тим щоб відстань від Ваших очей та екрану монітора була не менше 60-65 см а центр екрану знаходиться на рівні очей або трохи нижче;для уникнення "синдрому сухого ока" моргайте кожні 4 секунди; після роботи з ПК рекомендовано робити гімнастику для очей.

- Категорично не рекомендовано використовування телевізора замість монітору так як в ньому випромінювання в рази більше ніж монітору, тому що телевізор призначений для перегляду на суттєвій відстані.

- За будь-яких умов без перерви роботи за ПК дорослої людини не повинна перевищувати двох годин .

- При роботі з текстом рекомендується колір шрифту був темним, а колір фону – світлим при цьому шрифт повинен не бути занадто мілкий.

- Місце, де встановлений комп’ютер, щодня потрібно виконувати вологе прибирання;

- Приміщення де знаходиться комп’ютер треба провітрювати щогодини;

Вимоги безпеки під час роботи на ПК:

Потрібно слідити за станом екрану монітора: має бути чистим, без пилу та плям. Обов’язково слідкуйте за чистотою своїх окулярів – звичайних або комп’ютерних. Уникати контакту з проводами живлення і заземлення, задніх стінок комп’ютера чи кабелів. У разі виникнення аварійної ситуації необхідно негайно вимкнути ПК від електроенергії.

Вимоги щодо освітлення приміщень з ПК:

У приміщеннях освітлення повинно дотримуватись певних вимог: освітленість на робочому місці повинна відповідати зоровому характеру праці; повинні бути природне та штучне освітлення, також в межах простору робочого місця; на робочій поверхні не має бути різких тіней; у полі зору не повинно бути ніяких відблисків поверхні котрі викликають осліплення; вікна в приміщеннях переважно розташовані на Північ також Північний Схід які обладнані регульованими пристроями – занавіски, жалюзі і т.д.; рекомендовано обирати оптимальну спрямованість світлового потоку та склад світла. Освітленість робочої поверхні столу з ПК має становити не менше 400 ЛК, освітлення екрану не повинно перевищувати 200 ЛК. Як у разі штучного джерела світла, рекомендовано використовувати люмінесцентні лампи, використовування штучного освітлення без екрануючих решіток та розсіювачів не рекомендовано.

ВИСНОВКИ

Дана робота присвячена одній із найбільш актуальної теми сьогодення – база даних.

Можна сказати, що БД є найбільш ефективнішим механізмом зберігання й упорядкування даних, ніж електронні таблиці; це дає змогу створити централізований засіб, який можна легко модифікувати та швидко використовувати між декількома користувачів. Наявність веб-інтерфейсу знімає вимогу користувачів розуміти та використовувати базу даних безпосередньо, а також дозволяє користувачам підключатися з будь-якого місця за допомогою інтернету. Це також дає можливість запитам отримувати інформації для різних опитувань.

Розглянув такі поняття як БД, СУБД, модель даних та їх різновидність (концептуальна, логічна, фізична). Типи БД: мережна, ієрархічна, об’єктно-орієнтована та найбільш поширена у використанні реляційна. Розібрав зв’язки між таблицями: один – до – одного, один – до – багатьох, багато – до – одного, багато – до – багатьох.

Дослідив нормалізацію бази даних котра допомагає згрупувати пов’язані дані в таблицю та їхню нормальну форму (5NF).

Ознайомився з поняттям тригер, їх різновид, створення та видалення інструмента.

В цій роботі мною було побудовано ER-діаграму, створено та заповнено реляційну БД для офтальмологічної клініки “ЦЕНТР ЗОРУ” через додаток DB Browser (SQLite) , створював тригери до таблиць, для можливості коректного оброблення запитів на мові SQL .

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Гарсиа-Молина, Гектор, Ульман, Джеффри, Д., Уидом, Дженнифер. Г21 Системы баз данных. Полный курс. : Пер. с англ. — М. : Издательский дом. “Вильямс”, 2003.

[2] Lomet D., Salzberg B. Access Methods for Multiversion Data, Proc. 1989 ACM SIGMOD Int. Conf. on Management of Data, Portland, Ore. — May/June 1989.

[3] Knuth D.E. The Art of Computer Programming. Volume III: Sorting and Searching (2d ed.). Reading, Mass.: Addison-Wesley, 1998.

[4] Smith P.D., Barnes G. M. Files and Databases: An Introduction. Reading, Mass.: Addison-Wesley, 1987.

[5] Martin J. Computer Data-Base Organization (2d ed.). Englewood Cliffs, N.J.: Prentice-Hall, 1977.

[6] Реляційна модель баз даних - Бази даних.СУБД Access. Google Sites. URL: https://sites.google.com/site/bazidanihsubdaccess/bazi-danih-subd/ierarhicna-model-danih/relacijna-model-baz-danih

[7] Учасники проектів Вікімедіа. Моделі баз даних – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Моделі_баз_даних#:~:text=Основою%20бази%20даних%20є%20модель,мережева%20і%20реляційна%20моделі%20даних.

[8] Contributors to Wikimedia projects. Database model - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Database_model

[9] Database Models in DBMS | Studytonight. Studytonight - Best place to Learn Coding Online. URL: https://www.studytonight.com/dbms/database-model.php

[10] Database normalization description - Office. Developer tools, technical documentation and coding examples | Microsoft Docs. URL: https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description#:~:text=Normalization%20is%20the%20process%20of,eliminating%20redundancy%20and%20inconsistent%20dependency.

[11] Data Modelling: Conceptual, Logical, Physical Data Model Types. URL: https://www.guru99.com/data-modelling-conceptual-logical.html

[12] DBMS | Data Models - javatpoint. URL: https://www.javatpoint.com/data-models

[13] MySQL sample databases. Nanyang Technological University. URL: https://www3.ntu.edu.sg/home/ehchua/programming/sql/SampleDatabases.html

[14] Stages and Types of Data Models. TDAN.com. URL: https://tdan.com/stages-and-types-of-data-models/28201

[15] What is a database? Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/database/what-is-database/#:~:text=A%20database%20is%20an%20organized,database%20management%20system%20(DBMS).

[16] Relational Databases Basics. URL: https://learn.epam.com/detailsPage?id=c6b82e92-e019-4bf7-86c5-1e27fe23a1db.

[17] Генератор онлайн URL : https://generator-online.com/

[18] Генератор дати народження URL: https://rand.by/en/date

[19] Генератор телефонів онлайн URL : https://uk.calcprofi.com/henerator-vypadkovykh-nomeriv-telefonu.html

[20] Flowchart Maker & Online Diagram Software. Flowchart Maker & Online Diagram Software. URL: https://app.diagrams.net/

[21] Learn SQL: SQL Triggers. SQL Shack - articles about database auditing, server performance, data recovery, and more. URL: https://www.sqlshack.com/learn-sql-sql-triggers/