

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»

Кафедра інформаційних технологій

КУРСОВА РОБОТА

з дисципліни «Бази даних»

на тему: Проектування та розробка бази даних «Тестування знань»

Студента 3 курсу, групи ІПЗ-3
спеціальності Інженерія
програмного забезпечення
Полатайка І.Б.

Керівник старший викладач
Козич О.В.

Національна шкала: _____

Університетська шкала: _____

Оцінка ECTS: _____

Члени комісії:

_____ (підпис) _____ (прізвище та ініціали)

_____ (підпис) _____ (прізвище та ініціали)

_____ (підпис) _____ (прізвище та ініціали)

м. Івано-Франківськ – 2019 рік

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»

ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ

(прізвище, ім'я, по батькові студента)

Кафедра _____ Дисципліна _____

Спеціальність _____ Курс ____ Група ____ Семестр ____

1. Тема проекту_____

2. Рекомендована література _____

3. Перелік питань, які підлягають розробці _____

4. Дата видачі завдання _____

Термін подачі до захисту _____

5. Студент _____ Керівник _____

КАЛЕНДАРНИЙ ПЛАН

Полатайко І.Б.

(підпис)

Козич О В

(підпис)

РЕФЕРАТ

Пояснювальна записка: 55 сторінок (без додатків), 59 рисунків, 10 джерел, 1 додаток на 1 сторінці.

Ключові слова: БАЗА ДАНИХ, SQL, MYSQL, NOSQL, ER-ДІАГРАМА, ТЕСТУВАННЯ ЗНАНЬ, ЗАПИТ, ВИБІРКА, ЗОВНІШНІЙ КЛЮЧ, ПЕРВИННИЙ КЛЮЧ, ТАБЛИЦЯ, ЗАПИС.

Об'єктом дослідження є база даних «Тестування знань».

Мета роботи – розробити дизайн та продемонструвати реалізацію бази даних «Тестування знань».

Стислий опис тексту пояснювальної записки:

У даному курсовому проекті описано основні етапи проектування та розробки бази даних засобами MySQL, такі як: проектування БД у вигляді ER-діаграми, реалізація БД на основі діаграми, заповнення бази початковими даними та приклади роботи з готовою базою даних.

ABSTRACT

Explanatory note: 55 pages (without appendix), 59 figures, 10 references, 1 appendix on 1 page.

Keywords: DATABASE, SQL, MYSQL, NOSQL, ER-DIAGRAM, KNOWLEDGE TESTING, QUERY, SELECTION, FOREIGN KEY, PRIMARY KEY, TABLE, RECORD.

The object of study is the database "Testing knowledge".

The purpose of the work is to develop a design and demonstrate the implementation of the Knowledge Testing Database.

Brief description of the explanatory note text:

This course project describes the main stages of database design and development using MySQL, such as: designing a database in the form of an ER diagram, implementation of a database based on a diagram, filling the database with initial data and examples of working with resulting database.

ЗМІСТ

ВСТУП	7
1 ТЕОРІЯ БАЗ ДАНИХ.....	8
1.1 Визначення бази даних	8
1.2 Призначення баз даних	8
1.3 Різновиди баз даних	8
1.4 Relational SQL бази даних.....	9
1.5 NoSQL бази даних.....	11
2.1 Теорія проектування баз даних	14
2.2 Постановка проблеми	15
2.3 Аналіз проблеми.....	16
2.4 ER-діаграма бази даних	18
2.4.1 ER-діаграми	18
2.4.2 ER-діаграма бази даних «Тестування знань»	22
3 СТВОРЕННЯ БАЗИ ДАНИХ	24
3.1 Створення та дамп бази	24
3.2 Наповнення бази даними	26
4 ВИКОРИСТАННЯ БАЗИ ДАНИХ.....	32
4.1 Основні SQL запити та їх види.....	32
4.2 Виконання запитів у базі даних «Тестування знань».....	34
4.2.1 Демонстрація роботи INSERT запитів	34
4.2.4 Вибірки даних – SELECT	42
4.2.3 Оновлення даних – UPDATE.....	49
4.2.4 Видалення даних – DELETE.....	51
ВИСНОВКИ	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55
ДОДАТОК А	56

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-ХХ.ПЗ		
Розроб.	Полатайко І.Б.						
Перев.	Козич О.В.						
Н. контр.							
Затверд.							
Проектування та розробка бази даних «Тестування знань»					Літ.	Аркуш	Аркуши
					Н	6	55
					ПНУ ІПЗ-3		

ВСТУП

Бази даних є невід'ємною частиною майже будь-якого програмного продукту.

Дві основні задачі виникають при дизайні та розробці програмних рішень: обробка даних та збереження даних. Першу задачу вирішують програмні продукти. Другу ж – бази даних.

Завданням даної курсової роботи є розробка дизайну, а також практична реалізація бази даних для системи тестування знань. Системи тестування знань є досить поширеним типом програмних продуктів, які розробляються у більшості для навчальних закладів.

Наступні завдання повинні бути виконані в процесі роботи над дизайном та реалізацією бази даних:

- аналіз задачі та проблемної області
- розробка дизайну бази даних у вигляді ER діаграми
- створення та наповнення бази початковими даними
- демонстрація виконання базових запитів у даній базі даних

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
7						

1 ТЕОРІЯ БАЗ ДАНИХ

1.1 Визначення бази даних

База даних (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

1.2 Призначення баз даних

Головне завдання баз даних — гарантоване збереження значних обсягів інформації (так звані записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею.

Загалом бази використовуються для широкої сфери задач, де триває життя даних, повинна потенційно могти перевершити час роботи програми.

1.3 Різновиди баз даних

За типом збереження бази даних поділяються на два основних типи – це реляційні, також відомі, як SQL бази даних та нереляційні, також відомі як NoSQL бази даних. Різницю між ними розберемо далі. В загальному ж випадку реляційні бази даних надають спосіб збереження структурованих даних, та відрізняються більшою безпекою та точністю, в той час як NoSQL бази даних краще підходять для збереження неструктурованих або напівструктурованих даних та відрізняються кращою швидкістю роботи та більшою гнучкістю.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
8						

За фізичним способом збереження даних бази поділяються на централізовані (дані зберігаються в одному місці), розподілені (дані зберігаються розподілено) та хмарні (дані зберігаються розподілено у хмарних сервісах, забезпечуючи краще адміністрування та вищу безпеку даних).

1.4 Relational SQL бази даних

Реляційна база даних - це набір даних зі зв'язками між ними. Ці дані організовані у вигляді набору таблиць, що складаються із стовпців і рядків. У таблицях зберігається інформація про об'єкти, представлених в базі даних. У кожному стовпчику таблиці зберігається певний тип даних. Кожна стоку таблиці являє собою набір пов'язаних значень, що відносяться до одного об'єкту або сутності. Кожен рядок в таблиці позначається унікальним ідентифікатором, так званим первинним ключем, а рядки з декількох таблиць можуть бути пов'язані за допомогою зовнішніх ключів. До цих даних можна отримати доступ багатьма способами, і при цьому реорганізовувати таблиці БД не потрібно.

Реляційна база даних була винайдена в 1970 році Е. Ф. Коддом, тодішнім молодим програмістом в IBM. У своїй роботі "Реляційна модель даних для великих спільнот банків даних" Кодд запропонував перейти від зберігання даних в ієрархічних або навігаційних структурах до організації даних у таблицях, що містять рядки та стовпці.

Кожна таблиця, яку іноді називають відношенням, у реляційній базі даних містить одну або більше категорій даних у стовпцях або атрибутах. Кожен рядок, який також називається записом або кортежом, містить унікальний примірник даних або ключ для категорій, визначених стовпцями. Кожна таблиця має унікальний первинний ключ, який ідентифікує інформацію в таблиці. Відносини між таблицями можуть бути встановлені за допомогою використання зовнішніх ключів - полів в таблиці, які посилаються на первинний ключ іншої таблиці.

Наприклад, типова база даних для введення бізнес-замовлень міститиме таблицю, в якій описується клієнт із стовпцями для зберігання імені, адреси,

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
9						

номера телефону тощо. Ще одна таблиця описує замовлення: товар, клієнт, дата, ціна продажу тощо. Таким чином користувач реляційної бази даних може отримати уявлення про базу даних відповідно до своїх потреб. Наприклад, менеджеру відділення може знадобитися перегляд або звіт про всіх клієнтів, які придбали продукцію після певної дати. Менеджер фінансових послуг у тій же компанії може із тих же таблиць отримати звіт про рахунки, які потрібно сплатити.

Створюючи реляційну базу даних, ви можете визначити область можливих значень у стовпці даних та інші обмеження, які можуть застосовуватися до цього значення даних. Наприклад, домен можливих клієнтів може дозволити до 10 можливих імен клієнтів, але бути обмеженим в одній таблиці, щоб дозволити визначити лише три з цих імен клієнтів. Два обмеження стосуються цілісності даних та первинного та зовнішнього ключів:

Цілісність сутності забезпечує те, що первинний ключ у таблиці унікальний і що значення не встановлено на нульове значення.

Референтна цілісність вимагає, щоб кожне значення стовпця із зовнішнім ключем знаходилось у первинному ключі таблиці, з якої воно походить.

Приклади реляційних баз даних

Стандартні реляційні бази даних дозволяють користувачам керувати заздалегідь визначеними відносинами даних у кількох базах даних. Популярні приклади реляційних баз даних включають Microsoft SQL Server, Oracle Database, MySQL та IBM DB2.

Хмарні реляційні бази даних або бази даних як послуга (DBaaS) також широко використовуються, оскільки вони дають змогу компаніям передавати на аутсорсинг обслуговування баз даних, виправлення та підтримку інфраструктури, а також уникати потреби в датацентрів для фізичного зберігання даних. Хмарні реляційні бази даних включають службу Amazon Relational Database Service (RDS), Google Cloud SQL, IBM DB2 on Cloud, SQL Azure, Aurora DB, та Oracle Cloud.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
3м.	Арк.	№ докум.	Підпис	Дата		10

Переваги реляційних баз даних:

- Точність: дані зберігаються лише один раз, виключаючи дублікацію даних.
- Гнучкість: Складні запити для користувачів легко виконувати.
- Співпраця: Кілька користувачів можуть отримати доступ до однієї бази даних.
- Довіра: реляційні моделі баз даних зрілі та добре зрозумілі.
- Безпека: RDBMS підтримують ряд обмежень доступу для користувачів.

Основна перевага реляційних баз даних полягає в тому, що вони дозволяють користувачам легко класифікувати та зберігати дані, які згодом можуть бути отримані та відфільтровані для отримання конкретної інформації чи звітів. Реляційні бази даних також легко розширюються і не залежать від фізичної організації. Після створення оригінальної бази даних можна додати нову категорію даних без зміни всіх існуючих додатків.

1.5 NoSQL бази даних

NoSQL - це підхід до розробки баз даних, який включає такий спектр моделей даних, як ключ-значення, документо-орієнтована модель, стовпчикова модель та графова модель даних. NoSQL, який розшифровується як "не тільки SQL" ("Not only SQL"), є альтернативою традиційним реляційним базам даних, в яких дані розміщуються в таблицях, а схема даних ретельно розробляється перед тим, як створити базу даних. Бази даних NoSQL особливо корисні для роботи з великими наборами розподілених даних.

Термін NoSQL може бути застосований до деяких баз даних, які передували системі управління реляційними базами даних, але частіше це стосується баз даних, побудованих на початку 2000-х років з метою масштабного

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
11						

кластеризації баз даних у хмарних та веб-додатках. У цих додатках вимоги до продуктивності та масштабованості перевищували необхідність негайної, жорсткої узгодженості даних, яку RDBMS надав транзакційним корпоративним додаткам.

Зокрема, використовуючи NoSQL не потрібно було слідувати встановленій реляційній схемі. Великі веб-організації, такі як Google і Amazon, використовують NoSQL бази даних, для вузьких операційних цілях, а реляційні бази даних в якості допоміжних засобів, де необхідна повноцінна узгодженість даних.

Ранні бази даних NoSQL для веб та хмарних додатків, як правило, зосереджувались на дуже специфічних характеристиках управління даними. Можливість обробляти дуже великі обсяги даних та швидко розподіляти ці дані по обчислювальних кластерах були бажаними рисами веб-дизайну та хмарного дизайну. Розробники, які впроваджували хмарні та веб-системи, також намагалися створити гнучку схему даних - або взагалі ніякої схеми - для кращого включення швидких змін у додатки, які постійно оновлювалися.

Ключ-значення бази даних

У сховищах ключ-значення реалізована проста модель даних, яка поєднує унікальний ключ із пов'язаним значенням. Оскільки ця модель проста, вона дозволяє розробляти рішення, які є надзвичайно ефективними та широкомасштабними для управління сесіями та кешування в веб-додатках. Реалізації відрізняються тим, як вони орієнтовані на роботу з оперативною пам'яттю, твердотільними накопичувачами або дисководами. Приклади включають Dynamo DB, Aerospike, Berkeley DB, MemchacheDB, Redis і Riak.

Документо-орієнтовані бази даних

Бази даних документів, які також називаються сховищами документів, зберігають напівструктуровані дані та описи цих даних у форматі документа. Вони дозволяють розробникам створювати та оновлювати програми, не

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					KП.ІПЗ-03.ПЗ

потребуючи посилання на головну схему. Популярність цих баз даних зросла разом із зростом популярності JavaScript та JavaScript Object Notation (JSON), формату обміну даними, який набув широкого поширення серед розробників веб-додатків. Документо-орієнтовані бази даних використовуються для управління вмістом та обробки даних мобільних додатків. Couchbase Server, CouchDB, DocumentDB, MarkLogic та MongoDB - приклади таких баз даних.

Стовпчикові бази даних

Стовпчикові бази організують таблиці даних як стовпці, а не як рядки. Їх можна знайти як у базах даних SQL, так і в NoSQL. Вони можуть обробляти великі обсяги даних швидше, ніж звичайні реляційні бази даних. Широкий стовпчик зберігання даних може використовуватися для рекомендаційних механізмів, каталогів, антиспамінгових систем та інших видів обробки даних. Google BigTable, Cassandra та HBase - приклади стовпчикових баз даних.

Граф-орієнтовані бази даних

Граф-орієнтовані організовують дані у вигляді вузлів, подібних до записів у реляційній базі даних, і ребер, які представляють зв'язки між вузлами. Оскільки граф-орієнтована система зберігає взаємозв'язок між вузлами, вона може підтримувати складніші типи зв'язків. Крім того, на відміну від реляційних моделей, що спираються на суворі схеми, граф-модель даних дозволяє гнучно розширювати схему даних. Граф-орієнтовані бази даних застосовуються в системах, які повинні відображати взаємозв'язки, такі як системи бронювання або управління відносинами з клієнтами. Приклади баз даних графіків включають AllegroGraph, IBM Graph, Neo4j і Titan.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
13						

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1 Теорія проектування баз даних

Дизайн баз даних - це сукупність процесів, які полегшують проектування, розробку, впровадження та обслуговування систем управління даними підприємства. Правильно розроблена база даних проста в обслуговуванні, покращує узгодженість даних та економічно ефективна з точки зору відсутності дуплікації даних. Дизайнер бази даних вирішує, яким чином та які дані зберігати у базі, а також, які зв'язки встановити між цими даними.

Основними завданнями проектування баз даних є створення логічних та фізичних моделей конструкцій запропонованої системи баз даних.

Логічна модель концентрується на вимогах до даних та їх логічному представлення, не беручи до уваги модель фізичного збереження цих даних.

Фізична модель дизайну даних включає перехід від логічного проекту бази даних до розгляду способу зберігання даних на фізичний носій з використанням апаратних ресурсів та програмних систем, таких як системи управління базами даних (СУБД).

Проектування баз даних має вирішальне значення для високоефективної системи баз даних.

Зauważте, геніальність бази даних полягає в її розробці. Операції з використанням даних за допомогою SQL відносно прості

Процес проектування бази даних складається з кількох етапів, які послідовно слідують один за одним та в результаті приводять до результату, який являє собою модель бази даних.

Аналіз вимог

Планування - цей етап стосується планування всього циклу розробки баз даних. Він враховує стратегію інформаційних систем організації.

Визначення системи - цей етап визначає сферу та межі запропонованої системи баз даних.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
14						

Проектування баз даних

Логічна модель - цей етап стосується розробки моделі бази даних на основі вимог. Весь дизайн складається без будь-яких фізичних реалізацій або конкретних міркувань СУБД.

Фізична модель - Цей етап реалізує логічну модель бази даних з урахуванням СУБД та фізичних факторів реалізації.

Впровадження

Перетворення та завантаження даних - цей етап стосується імпорту та фізичної реалізації бази даних.

Тестування - цей етап стосується виявлення помилок у нещодавно впроваджений системі. Він перевіряє базу даних на предмет відповідності до специфікацій та вимог.

Два типи завдань при проектуванні бази даних:

- Нормалізація
- ER моделювання

2.2 Постановка проблеми

Темою даної курсової роботи є розробка бази даних, призначеної для системи тестування знань.

Системи електронного тестування знань стають все популярнішими і плавно витісняють інші, раніше прийняті способи контролю знань.

Перевагами систем електронного тестування знань є:

- централізоване збереження інформації про результати тестувань
- швидкий доступ до результатів тестів певної особи
- збільшення гнучкості проходження тестів
- зменшення ризиків пов'язаних з втратою інформації про результати студентів

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
15						

Проте, в той самий час, коли системи електронного тестування знань вирішують багато проблем з гнучкістю тестування та централізованим збереженням результатів до таких систем ставляться ряд вимог, пов'язаних з безпекою збереження даних, а саме: зведення до мінімуму можливостей втрати чи модифікації даних, шляхом несанкціонованого доступу чи технічних несправностей.

2.3 Аналіз проблеми

Під час аналізу проблемної області було вирішено, що база даних буде дозволяти зберігати наступні види запитань, а також відповідей студентів:

- тестові запитання (одна відповідь)
- тестові запитання (кілька відповідей)
- запитання з відкритою відповіддю
- запитання на сполучення відповідей.

А також було виокремлено наступні дійові особи:

- студент (будь-яка людина, яка буде здавати тести в системі)
- модератор (закріплений за групою, може реєструвати групу на певні курси)

Протягом періоду аналізу проблеми було прийнято наступні рішення, що вплинули на ті, які були прийняті на етапі проектування:

- студент може зареєструватися в системі, після чого йому стають доступними курси на які він може також зареєструватися (прийнято, що всі курси є вільними для реєстрації)
- після реєстрації на певний курс, студенту стають доступними тести, які він може здавати
- після здачі всіх тестів з певного курсу, вважається, що студент завершив курс

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
16						

- студент може бути прикріпленим до довільної кількості груп, в такому випадку, він може бути зареєстрований на курс також модератором тієї групи, до якої він прикріплений.
- студент має набір тестів, доступних для здачі, туди входять: тести, що належать до курсів, на які зареєстрований студент, тести, на які студент незалежно зареєстрований.

Для розробки цієї бази було вибрано технологію SQL, оскільки моделі даних в нашій базі добре структуруються. З реалізації SQL було вибрано MySQL, оскільки це реалізація є досить пошириною та задовольняє вимоги до СУБД, які передбачає дизайн даної бази даних.

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					17

2.4 ER-діаграма бази даних

2.4.1 ER-діаграми

ER або (Реляційна модель особи) - це концептуальна модель даних високого рівня. Модель відношення базується на понятті суб'єктів реального світу та взаємозв'язку між ними.

Діаграма відносин (ER) - це тип блок-схеми, що ілюструє, як "сущності", такі як люди, об'єкти чи поняття, співвідносяться один з одним у системі. ER-діаграми найчастіше використовуються для проектування реляційних баз даних у галузі програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Також відомі як ERD або ER-моделі, вони використовують визначений набір символів для відображення взаємозв'язку сущностей, відносин та їх атрибутів.

Моделювання ER допомагає систематично аналізувати вимоги до даних, щоб створити добре спроектовану базу даних. Найкращою практикою є завершення моделювання ER перед впровадженням вашої бази даних.

Діаграми ER - це наочний інструмент, який корисний для представлення моделі ER. Петром Ченом було запропоновано в 1971 р. створити єдину конвенцію, яка може використовуватися для реляційних баз даних та мережі. Він мав на меті використовувати модель ER як концептуальний підхід до моделювання.

Діаграма відносин відображає відносини сукупності об'єктів, що зберігаються в базі даних. Іншими словами, можна сказати, що діаграми ER допомагають пояснити логічну структуру бази даних. На перший погляд, діаграма ER виглядає дуже схоже на блок-схему. Однак діаграма ER включає багато спеціалізованих символів, і їх значення роблять цю модель унікальною.

Документуючи систему чи процес, розгляд системи декількома способами збільшує розуміння цієї системи. ЕРД-діаграми зазвичай використовуються разом із діаграмою потоку даних для відображення вмісту сховища даних. Вони

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
18						

допомагають нам візуалізувати, як дані пов'язані загальним чином, і особливо корисні для побудови реляційної бази даних.

Факти про ER діаграми:

- модель ER дозволяє наочно зобразити дизайн бази даних
- це простий у використанні графічний інструмент для моделювання даних
- широко використовується в дизайні баз даних
- це графічне представлення логічної структури бази даних
- допомагає визначити сутності, які існують у системі, та відносини між цими сутностями

Основні причини використання ER діаграми:

- допомагає визначити терміни, пов'язані з моделюванням відносин з сутністю
- демонструє попередній перегляд того, як повинні з'єднуватися всі ваші таблиці, які поля будуть у кожній таблиці
- допомагає описати сутності, ознаки, відносини
- на основі ER-діграми можна створити код генерації бази даних
- ER діаграми можуть бути використані дизайнерами баз даних як документація для реалізації даних у конкретних застосунках
- дизайнер бази даних отримує краще розуміння інформації, яка міститься в базі даних

Компоненти ER діаграми:

Ця модель базується на трьох основних поняттях:

- сутності
- атрибути
- відношення

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
19						

Сутність:

Об'єкт, який можна легко виділити.

Атрибут:

Певна характеристика сутності. Наприклад, ім'я є характеристикою певного студента.

Відношення:

Відношення - це асоціація між двома або більше сутностями.

Визначає числові атрибути взаємозв'язку між двома сутностями або множинами сутностей.

Різними типами відношень є:

- один до одного
- один до багатнього
- багато до одного
- багато до багатьох

Відношення типу один-до-одного означає, що один примірник першої сутності (лівої) пов'язаний з одним примірником другої сутності (правої). Відношення один-до-одного найчастіше свідчить про те, що насправді ми маємо всього одну сутність, розділену на дві.

Відношення типу один-до-багатьох означає, що один примірник першої сутності (лівої) пов'язаний з декількома екземплярами другої сутності (правої). Це найбільш часто використовуваний тип зв'язку. Ліва сутність (з боку "один") називається батьківською, права (з боку "багато") – дочірньою. Відношення багато-до-одного є тим самим відношенням, проте дзеркально відображенім.

Відношення типу багато-до-багатьох означає, що кожен екземпляр першої сутності може бути пов'язаний з декількома екземплярами другої сутності, і кожен примірник другої сутності може бути пов'язаний з декількома примірниками першої сутності.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
20						

Позначення зв'язків:

На рисунках 2.1-2.4 вказано позначення основних типів зв'язків у ER діаграмах.



Рисунок 2.1 – Строго один



Рисунок 2.2 – Нуль або один

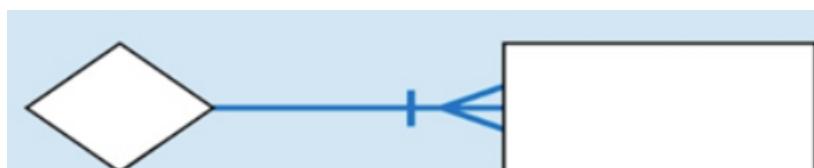


Рисунок 2.3 – Один або багато

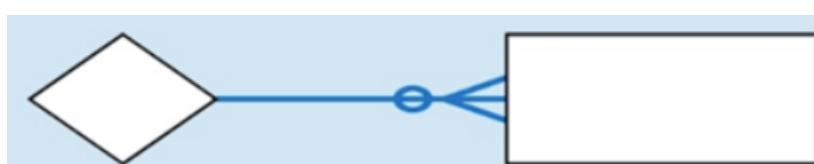


Рисунок 2.4 – Нуль або багато

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
21						

2.4.2 ER-діаграма бази даних «Тестування знань»

Після проведеного аналізу було розроблено наступну ER-діаграму бази даних.

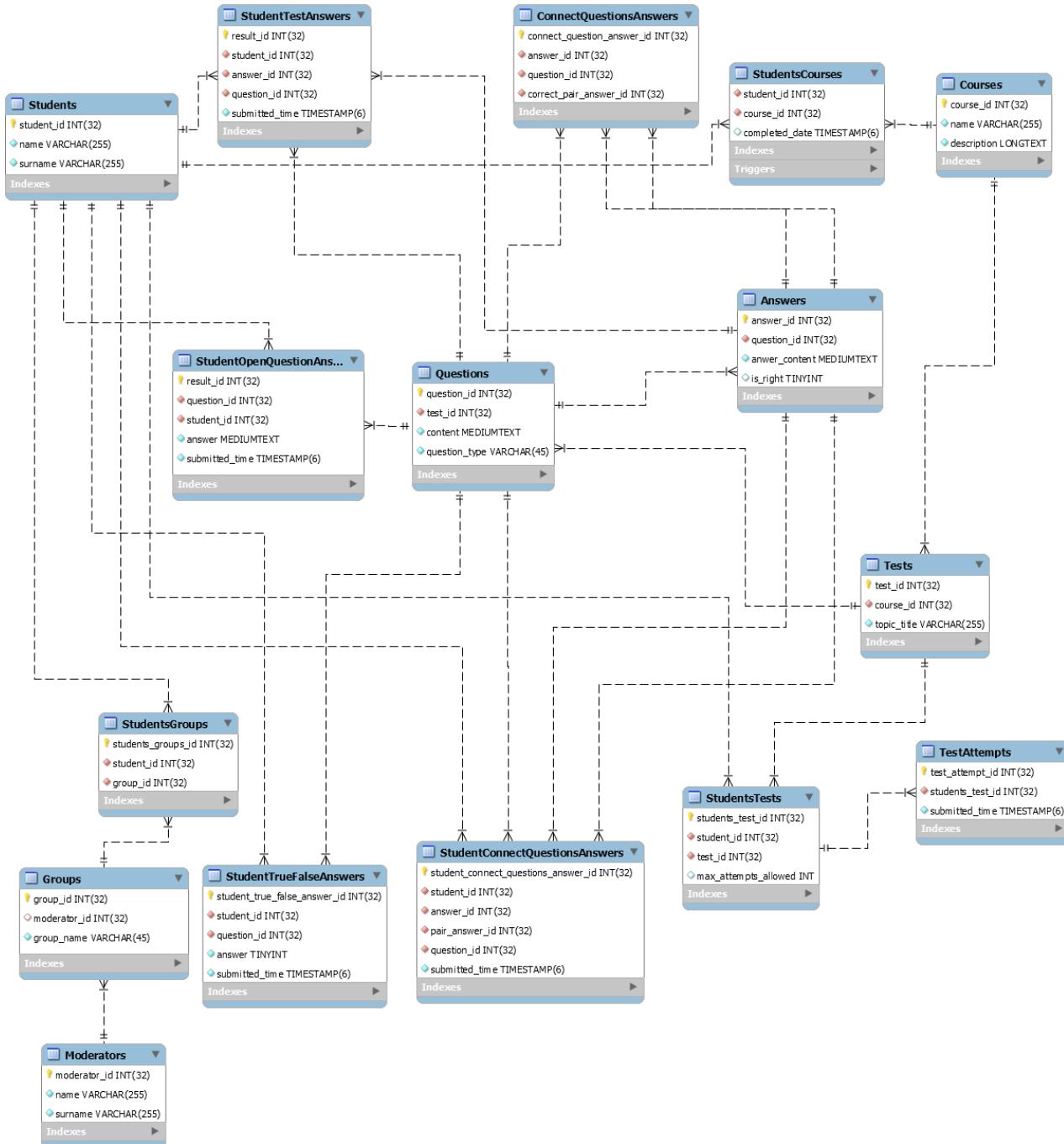


Рисунок 2.5 – ER-діаграма

Зв’язки між таблицями подано на діаграмі.

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					22

База включатиме в себе два тригера, які спрацьовуватимуть тоді, коли студент буде реєструватися, або відмінити реєстрацію на певний курс. Їхня задача – зберігати таблиці StudentsTests та StudentsCourses в синхронізованому стані.

Тригер, що спрацьовуватиме після додавання даних, до таблиці StudentsCourses:

```
DELIMITER ;;

CREATE TRIGGER `StudentsCourses_AFTER_INSERT` AFTER INSERT ON
`studentscourses` FOR EACH ROW BEGIN

    INSERT INTO StudentsTests(student_id, test_id) (SELECT
NEW.student_id, test_id FROM Tests WHERE course_id =
NEW.course_id);

END ;;

DELIMITER ;
```

Тригер, що спрацьовуватиме після видалення даних з таблиці StudentsCourses:

```
DELIMITER ;;

CREATE TRIGGER `StudentsCourses_AFTER_DELETE` AFTER DELETE ON
`studentscourses` FOR EACH ROW BEGIN

    DELETE FROM StudentsTests WHERE test_id IN (SELECT test_id
FROM Tests AS t WHERE t.course_id = OLD.course_id);

END ;;

DELIMITER ;
```

Зм.	Арк.	№ докум.	Підпис	Дата

3 СТВОРЕННЯ БАЗИ ДАНИХ

3.1 Створення та дамп бази

На основі створеної ER діаграми (рис 2.5), згенеруємо SQL код, за допомогою якого можемо створити нашу базу (Додаток Б). Такий код називається *дампом структури бази даних*. Його можна згенерувати за допомогою середовища розробки ER діаграми, яким в даному випадку був MySQL WorkBench.

Отож, з отриманого коду створимо базу даних, просто імпортувавши його та виконавши.

The screenshot shows the MySQL Workbench interface. The top part is the SQL editor window, displaying the generated SQL script. The script includes comments indicating it was generated by MySQL Workbench on December 15, 2019, at 03:03:35. It sets session variables for unique checks, foreign key checks, and SQL mode, then creates a schema named 'KnowledgeTestingDb' and selects it. Finally, it creates a table 'Moderators' with columns 'id' (primary key) and 'name'. The table has a foreign key constraint 'moderator_id' referencing the primary key 'id' in the 'Moderators' table of the same schema. The script concludes with setting the engine to InnoDB. The bottom part is the Output pane, which shows the results of the execution: 10 rows affected for each of the six statements (CREATE SCHEMA, USE, CREATE TABLE, and the three SET commands).

```
-- MySQL Script generated by MySQL Workbench
-- Sun Dec 15 03:03:35 2019
-- Model: New Model  Version: 1.0
-- MySQL Workbench Forward Engineering

1 -- MySQL Script generated by MySQL Workbench
2 -- Sun Dec 15 03:03:35 2019
3 -- Model: New Model  Version: 1.0
4 -- MySQL Workbench Forward Engineering
5
6 • SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7 • SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
8 • SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_NAME_CHECK';
9
10 -----
11 -- Schema KnowledgeTestingDb
12 -----
13
14 -----
15 -- Schema KnowledgeTestingDb
16 -----
17 • CREATE SCHEMA IF NOT EXISTS `KnowledgeTestingDb` DEFAULT CHARACTER SET utf8 ;
18 • USE `KnowledgeTestingDb` ;
19
20 -----
21 -- Table `KnowledgeTestingDb`.`Moderators`
```



```
40   CONSTRAINT `moderator_id`
41     FOREIGN KEY (`moderator_id`)
42       REFERENCES `KnowledgeTestingDb`.`Moderators` (`moderator_id`)
43       ON DELETE NO ACTION
44       ON UPDATE NO ACTION
45   ENGINE = InnoDB;
```

Output:

#	Time	Action	Message
405	03:34:48	CREATE DEFINER = CURRENT_USER TRIGGER 'KnowledgeTestingDb'.`StudentsCourses_AFTER_INSERT`	0 row(s) affected
406	03:34:48	USE 'KnowledgeTestingDb'	0 row(s) affected
407	03:34:48	CREATE DEFINER = CURRENT_USER TRIGGER 'KnowledgeTestingDb'.`StudentsCourses_AFTER_DELETE`	0 row(s) affected
408	03:34:48	SET SQL_MODE=@OLD_SQL_MODE	0 row(s) affected
409	03:34:48	SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS	0 row(s) affected
410	03:34:48	SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS	0 row(s) affected

Рисунок 3.1 – Створення бази даних

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						24

Загалом, запити, які виконуються в даному випадку відносяться до підмножини DDL (data definition language – мова опису даних) мови запитів SQL. Вона використовується для того, що задати або змінити структурний вигляд бази даних.

До операторів DDL відносяться наступні:

- Create (Створити)
 - Alter (Змінити)
 - Drop (Видалити)

Тепер за допомогою команди SHOW TABLES виведемо список всіх таблиць з нашої бази даних.

Як бачимо база успішно створена.

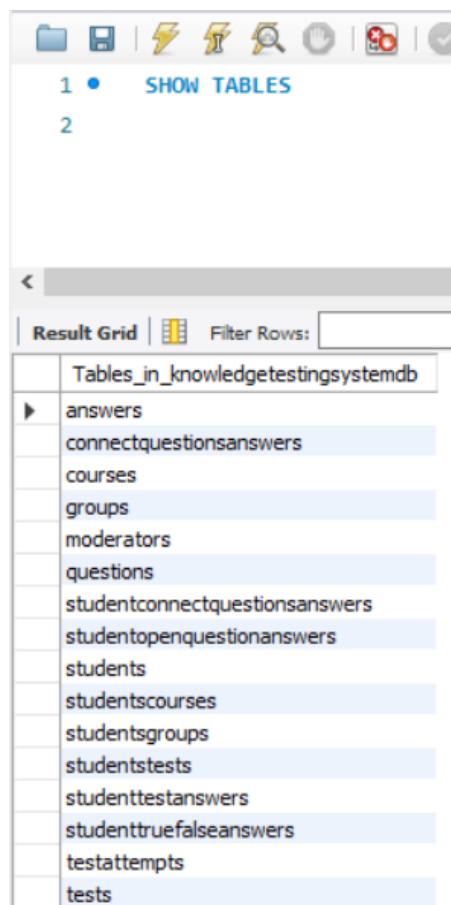


Рисунок 3.2 – Список таблиць базы данных

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ

3.2 Наповнення бази даними

Тепер, коли наша база створена перейдемо до наступного етапу, а саме етапу заповнення нашої бази даними.

Для початку заповнимо таблицю, яка містить інформацію про студентів:

The screenshot shows the MySQL Workbench interface. In the top pane, there is an SQL editor window containing the following code:

```
1 • INSERT INTO students(name, surname) VALUES('Ihor', 'Polataiko');
2 • INSERT INTO students(name, surname) VALUES('Ed', 'Miller');
3 • INSERT INTO students(name, surname) VALUES('John', 'Brain');
4 • INSERT INTO students(name, surname) VALUES('Martin', 'Fowler');
5 • INSERT INTO students(name, surname) VALUES('Mary', 'Williams');
6 • INSERT INTO students(name, surname) VALUES('Kate', 'Devis');
7 • INSERT INTO students(name, surname) VALUES('Ann', 'Anderson');
8 • INSERT INTO students(name, surname) VALUES('Antony', 'Joshua');
9 • INSERT INTO students(name, surname) VALUES('Kate', 'Sierra');
10 • INSERT INTO students(name, surname) VALUES('Alan', 'Brown');
11 • INSERT INTO students(name, surname) VALUES('Alice', 'Liddell');
12 • INSERT INTO students(name, surname) VALUES('Dale', 'Smith');
```

In the bottom pane, there is an "Output" window titled "Action Output" showing the results of the execution:

#	Time	Action	Message
255	04:19:11	INSERT INTO students(name, surname) VALUES('Ihor', 'Polataiko')	1 row(s) affected
256	04:19:11	INSERT INTO students(name, surname) VALUES('Ed', 'Miller')	1 row(s) affected
257	04:19:11	INSERT INTO students(name, surname) VALUES('John', 'Brain')	1 row(s) affected
258	04:19:11	INSERT INTO students(name, surname) VALUES('Martin', 'Fowler')	1 row(s) affected
259	04:19:11	INSERT INTO students(name, surname) VALUES('Mary', 'Williams')	1 row(s) affected
260	04:19:11	INSERT INTO students(name, surname) VALUES('Kate', 'Devis')	1 row(s) affected
261	04:19:11	INSERT INTO students(name, surname) VALUES('Ann', 'Anderson')	1 row(s) affected
262	04:19:11	INSERT INTO students(name, surname) VALUES('Antony', 'Joshua')	1 row(s) affected
263	04:19:11	INSERT INTO students(name, surname) VALUES('Kate', 'Sierra')	1 row(s) affected
264	04:19:11	INSERT INTO students(name, surname) VALUES('Alan', 'Brown')	1 row(s) affected
265	04:19:11	INSERT INTO students(name, surname) VALUES('Alice', 'Liddell')	1 row(s) affected
266	04:19:11	INSERT INTO students(name, surname) VALUES('Dale', 'Smith')	1 row(s) affected

Рисунок 3.3 – Заповнення таблиці Students

Також створимо кілька груп, що буде потрібно про подальшій демонстрації запитів.

The screenshot shows the MySQL Workbench interface. In the top pane, there is an SQL editor window containing the following code:

```
1 • INSERT INTO `groups`(group_name) VALUE('developers');
2 • INSERT INTO `groups`(group_name) VALUE('lifetime -learners');
```

In the bottom pane, there is an "Output" window titled "Action Output" showing the results of the execution:

#	Time	Action	Message
1	04:26:10	INSERT INTO `groups`(group_name) VALUE('developers')	1 row(s) affected
2	04:26:10	INSERT INTO `groups`(group_name) VALUE('lifetime -learners')	1 row(s) affected

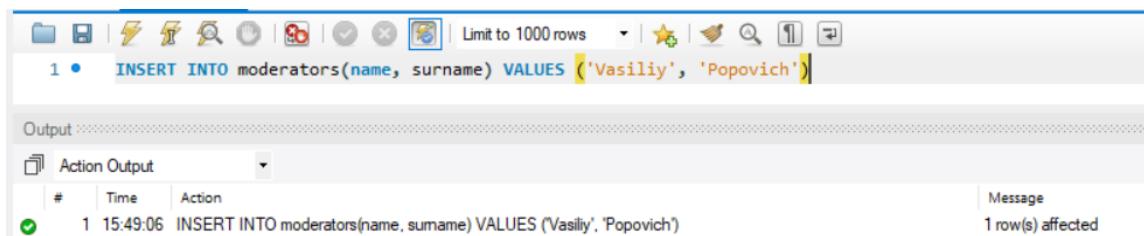
Рисунок 3.4 – Заповнення таблиці Groups

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						26

Також, відмітимо, що слово `groups` є ключовим словом мови SQL, тому для того, щоб використати його як назву таблиці при формуванні запиту його порібно взяти в лапки: `groups`.

Використання ключових слів на найменування таблиць є поганою практикою і в даній роботі приведено тільки для ілюстрації формування запиту, що містить ключові слова у ролі назв таблиць чи атрибутив.

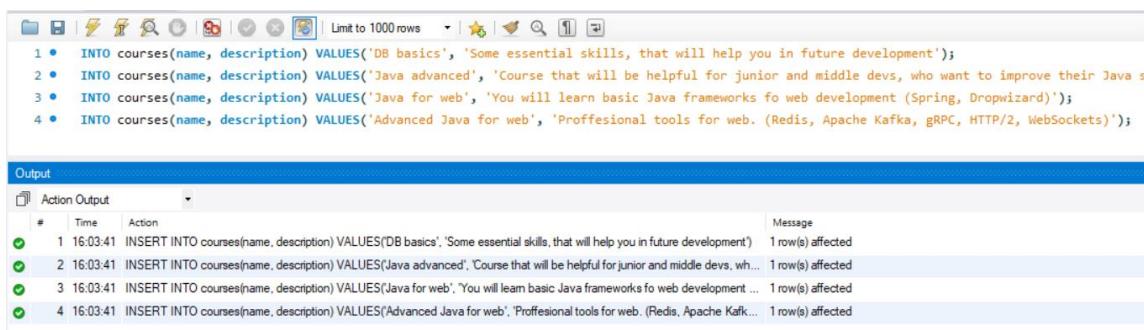
Також додамо запис до таблиці Moderators, для подальшого використання.



The screenshot shows the MySQL Workbench interface. In the top query editor, there is a single line of SQL code: `INSERT INTO moderators(name, surname) VALUES ('Vasiliy', 'Popovich')`. Below the editor, the 'Output' tab is selected, showing the results of the query. A table titled 'Action Output' contains one row: # 1, Time 15:49:06, Action `INSERT INTO moderators(name, surname) VALUES ('Vasiliy', 'Popovich')`, and Message `1 row(s) affected`.

Рисунок 3.5 – Заповнення таблиці Moderators

Тепер перейдемо до таблиці, які міститиме інформацію про курси.



The screenshot shows the MySQL Workbench interface. The top query editor contains four separate `INSERT INTO courses` statements. The bottom part of the interface shows the 'Output' tab with the results of these queries. A table titled 'Action Output' lists four rows, each corresponding to one of the inserted records. Each row includes a timestamp (16:03:41), the action (an `INSERT INTO` statement), and a message indicating '1 row(s) affected'.

Рисунок 3.6 – Заповнення таблиці Courses

Наступною заповнимо таблицю, яка містить інформацію про тести, що які будуть доступні у нашій системі.

Зверніть увагу, що у даній таблиці ми використовуємо зовнішній ключ (foreign key) для того, щоб зв'язати між собою таблиці Courses та Tests. Таким

Зм.	Арк.	№ докум.	Підпис	Дата

чином кожен тест буде прикріпленим до певного курсу. Зовнішнім ключем в даному випадку є поле course_id у таблиці Tests.

```

1 •  INSERT INTO tests(course_id, topic_title) VALUES(5, 'What is db?');
2 •  INSERT INTO tests(course_id, topic_title) VALUES(5, 'Query language');
3 •  INSERT INTO tests(course_id, topic_title) VALUES(5, 'Relations');
4 •  INSERT INTO tests(course_id, topic_title) VALUES(6, 'Connection to sockets');
5 •  INSERT INTO tests(course_id, topic_title) VALUES(6, 'Connection to DB');
6 •  INSERT INTO tests(course_id, topic_title) VALUES(6, 'JMX tools');
7 •  INSERT INTO tests(course_id, topic_title) VALUES(7, 'Spring MVC essentials');
8 •  INSERT INTO tests(course_id, topic_title) VALUES(7, 'Web architecture');
9 •  INSERT INTO tests(course_id, topic_title) VALUES(7, 'DropWizard');
10 •  INSERT INTO tests(course_id, topic_title) VALUES(8, 'gRPC, HTTP/2');
11 •  INSERT INTO tests(course_id, topic_title) VALUES(8, 'Apache Kafka');

Output
Action Output
# Time Action Message
1 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(5, 'What is db?') 1 row(s) affected
2 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(5, 'Query language') 1 row(s) affected
3 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(5, 'Relations') 1 row(s) affected
4 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(6, 'Connection to sockets') 1 row(s) affected
5 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(6, 'Connection to DB') 1 row(s) affected
6 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(6, 'JMX tools') 1 row(s) affected
7 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(7, 'Spring MVC essentials') 1 row(s) affected
8 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(7, 'Web architecture') 1 row(s) affected
9 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(7, 'DropWizard') 1 row(s) affected
10 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(8, 'gRPC, HTTP/2') 1 row(s) affected
11 16:22:24 INSERT INTO tests(course_id, topic_title) VALUES(8, 'Apache Kafka') 1 row(s) affected

```

Рисунок 3.7 – Заповнення таблиці Tests

В загальному випадку foreign key використовується для побудови зв’язків між таблицями, а також для забезпеченням цілісності даних, оскільки в дію вступають певні обмеження та правила синхронізації.

Наприклад, при видаленні запису первинний ключ якого є вторинним ключем іншого з метою забезпечення цілісності даних SQL надає три можливих варіанти розвитку подій:

- буде також видалено запис з вторинним ключем
- видалення запису з первинним ключем буде неможливим, доки існуватиме запис з вторинним ключем
- видалення відбудеться, але у полі вторинного ключа буде проставлено значення

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						28

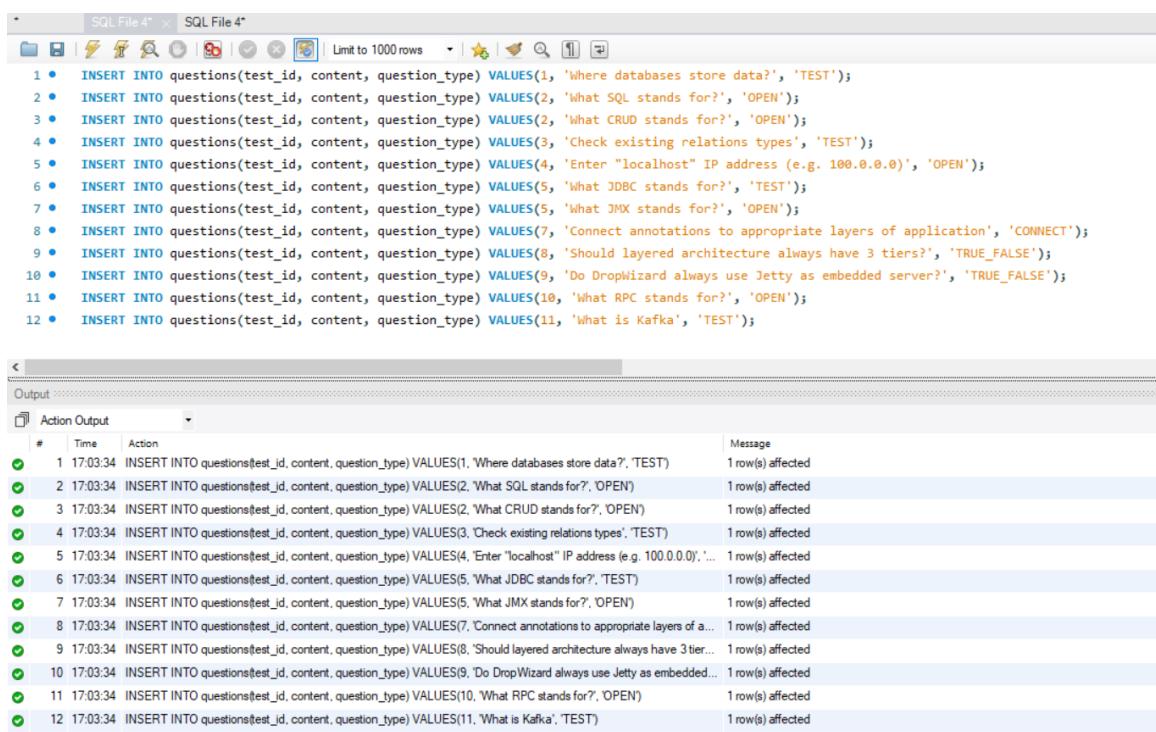
А також при оновленні первинного ключа, всі вторинні ключі також будуть змінені для забезпечення збереження зв'язку.

Тепер перейдемо до таблиці Questions.

Відповідно до дизайну бази ця таблиця містить поле question_type, яке визначає тип запитання та може містити тільки чотири можливі значення. Для того, щоб забезпечити цілісність даних у такому випадку ми використовуємо обмеження можливих значень для даного поля.

```
CONSTRAINT `question_type` CHECK (question_type IN ('TRUE_FALSE', 'TEST', 'OPEN', 'CONNECT'))
```

Рисунок 3.8 – Додавання обмеження на поле



```
CONSTRAINT `question_type` CHECK (question_type IN ('TRUE_FALSE', 'TEST', 'OPEN', 'CONNECT'))
```

Рисунок 3.9 – Заповнення таблиці Questions

Зм.	Арк.	№ докум.	Підпис	Дата

Заповнимо таблицю Answers даними.

Зверніть увагу, що поля які можуть приймати значення NULL вказувати не обов'язково, в такому випадку вони будуть заповнені значенням NULL автоматично.

Це саме стосується також і полів які мають значення за замовчуванням (DEFAULT), але в цьому випадку, якщо значення не вказано, то поле буде заповнене значенням за замовчуванням.

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains 27 numbered SQL INSERT statements for the 'answers' table. The columns are 'question_id' and 'answer_content'. The values range from 'Structured Query Language' to 'Database'.
- Output Panel:** Shows the results of the executed queries. It includes a table titled 'Action Output' with columns '#', 'Time', 'Action', and 'Message'. The log shows three successful insert operations:

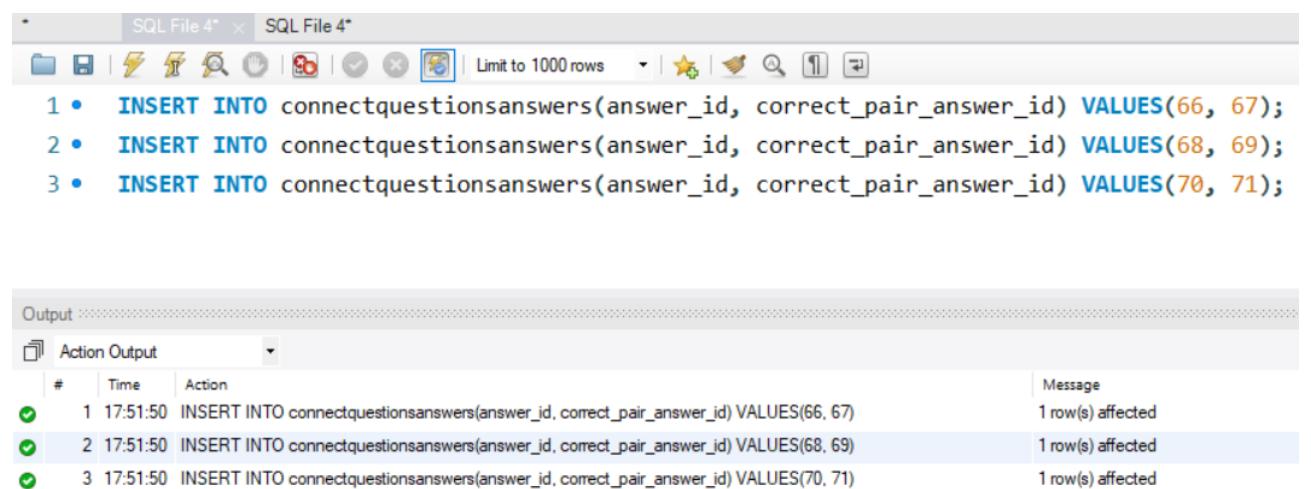
#	Time	Action	Message
25	17:48:29	INSERT INTO answers(question_id, answer_content, is_right) VALUES(12, 'Distributed event streaming platform', true)	1 row(s) affected
26	17:48:29	INSERT INTO answers(question_id, answer_content, is_right) VALUES(12, 'Message queue', false)	1 row(s) affected
27	17:48:29	INSERT INTO answers(question_id, answer_content, is_right) VALUES(12, 'Database', false)	1 row(s) affected

Рисунок 3.10 – Заповнення таблиці Answers

Наочник, додамо записи до таблиці ConnectQuestionAnswers, що потрібна для того, щоб зберігати пари правильних відповідей у завданнях, які вимагають з'єднання варіантів відповідей.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						30

Даний дизайн таблиці дозволяє зберігати поділ відповідей на групи, що важливо для такого типу запитань, оскільки відповідям з лівого стовпця повинні ставитися у відповідність тільки відповіді з правого (і навпаки).



The screenshot shows the MySQL Workbench interface with two tabs open: "SQL File 4*" and "Output". The SQL tab contains three INSERT statements:

```
1 • INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(66, 67);
2 • INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(68, 69);
3 • INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(70, 71);
```

The Output tab displays the results of these statements:

#	Time	Action	Message
1	17:51:50	INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(66, 67)	1 row(s) affected
2	17:51:50	INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(68, 69)	1 row(s) affected
3	17:51:50	INSERT INTO connectquestionsanswers(answer_id, correct_pair_answer_id) VALUES(70, 71)	1 row(s) affected

Рисунок 3.11 – Заповнення таблиці ConnectQuestionAnswers

4 ВИКОРИСТАННЯ БАЗИ ДАНИХ

4.1 Основні SQL запити та їх види

SQL запити – інструменти мови SQL, які призначені для виконання операцій над даними, структурою бази даних чи самою базою даних.

Ось список основних видів SQL запитів:

- DDL – Data Definition Language (мова визначення даних)
- DQL – Data Query Language (Мова запиту даних)
- DML – Data Manipulation Language (Мова маніпуляції даними)
- DCL – Data Control Language (Мова управління даними)
- TCL – Transaction Control Language (Мова контролю транзакцій)

Розглянемо їх детальніше.

DDL (Мова визначення даних):

DDL або мова визначення даних фактично складається з команд SQL, які можна використовувати для визначення схеми бази даних. Вона стосується опису схеми бази даних і використовується для створення та модифікації об'єктів бази даних.

Приклади команд DDL:

- CREATE - використовується для створення бази даних або її об'єктів (наприклад, таблиця, індекс, функція, представлення даних, процедура зберігання та тригери).
- DROP - використовується для видалення об'єктів бази даних.
- ALTER - використовується для зміни структури бази даних.
- TRUNCATE - використовується для видалення всіх записів із таблиці.
- COMMENT - використовується для додавання коментарів до словника даних.
- RENAME - використовується для перейменування об'єкта, що існує в базі даних.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
32						

DQL (Мова запиту даних):

Мова запитів DQL використовуються для виконання запитів для вибірки даних.

Приклад DQL:

- SELECT - використовується для отримання даних із бази даних.

DML (Мова маніпуляції даними):

Команди SQL, які займаються маніпулюванням даними, наявними в базі, належать до DML або мови маніпуляції даними.

Приклади DML:

- INSERT - використовується для вставки даних у таблицю.
- UPDATE - використовується для оновлення наявних даних у межах таблиці.
- DELETE - використовується для видалення записів із таблиці бази даних.

DCL (Мова управління даними):

DCL включає команди, такі як GRANT та REVOKE, які в основному стосуються прав, дозволів та інших елементів управління базою даних.

Приклади команд DCL:

- GRANT - надає користувачеві права доступу до бази даних.
- REVOKE - вилучає права доступу у користувача, надані за допомогою команди GRANT.

TCL (Мова контролю транзакцій):

Команди TCL займаються транзакціями в межах бази даних.

Приклади команд TCL:

- COMMIT - затверджує транзакцію.
- ROLLBACK - відміняє транзакцію у випадку виникнення будь-якої помилки.
- SAVEPOINT - встановлює точку збереження в межах транзакції.
- SET TRANSACTION - задайте характеристики транзакції.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
33						

4.2 Виконання запитів у базі даних «Тестування знань»

4.2.1 Демонстрація роботи INSERT запитів

Запити для додавання даних вже використовувалися у цій курсовій роботі при початковому заповненні бази даних.

Продовжимо заповнення нашої бази даними зареєструвавши кількох студентів на курс.

The screenshot shows the MySQL Workbench interface. In the top query editor, four INSERT statements are listed:

```
1  INSERT INTO studentscourses(student_id, course_id) VALUES(1, 8);
2 • INSERT INTO studentscourses(student_id, course_id) VALUES(3, 5);
3 • INSERT INTO studentscourses(student_id, course_id) VALUES(6, 6);
4 • INSERT INTO studentscourses(student_id, course_id) VALUES(12, 7);
```

In the bottom 'Output' pane, the results of these statements are shown in a table:

Action	Time	Action	Message
1	08:28:29	INSERT INTO studentscourses(student_id, course_id) VALUES(1, 8)	1 row(s) affected
2	08:28:29	INSERT INTO studentscourses(student_id, course_id) VALUES(3, 5)	1 row(s) affected
3	08:28:29	INSERT INTO studentscourses(student_id, course_id) VALUES(6, 6)	1 row(s) affected
4	08:28:29	INSERT INTO studentscourses(student_id, course_id) VALUES(12, 7)	1 row(s) affected

Рисунок 4.1 – Запис студентів на курс

Зверніть увагу, що під час проведення такої процедури, також спрацював тригер, який повинен зберігати таблиці StudentsCourses та StudentsTests в синхронізованому стані.

Тригери застосовуються для забезпечення цілісності даних і реалізації складної бізнес-логіки. Тригер запускається сервером автоматично при спробі зміни даних у таблиці, з якою він пов'язаний. Всі здійснені ним модифікації даних розглядаються як виконані в транзакції, в якій виконано дію, що викликало спрацьовування тригера. Відповідно, у разі виявлення помилки або порушення цілісності даних може відбутися відкат цієї транзакції.

Таким чином тепер до таблиці StudentsTests додалися записи про реєстрацію студентів на здачу певних тестів, відповідно до курсів, на які вони були зареєстровані.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						34

Отримаємо вибірку даних з таблиці, щоб переконатися в цьому.

The screenshot shows the MySQL Workbench interface. In the top query editor, the SQL command `SELECT * FROM studentstests` is entered. Below it, the 'Result Grid' shows the following data:

	students_test_id	student_id	test_id	completed_date
▶	1	1	10	NULL
	2	1	11	NULL
	4	3	1	NULL
	5	3	2	NULL
	6	3	3	NULL
	7	6	4	NULL
	8	6	5	NULL
	9	6	6	NULL
	10	12	7	NULL
	11	12	8	NULL
	12	12	9	NULL
*	NULL	NULL	NULL	NULL

Рисунок 4.2 – Демонстрація роботи тригера

Відповідно до дизайну бази студент може бути зареєстрованим на певний тест не будучи зареєстрованим на курс. Продемонструємо це наступним запитом, зареєструвавши студента з ідентифікатором 1 (Ihor Polataiko) на тест з ідентифікатором 8 (Web architecture).

The screenshot shows the MySQL Workbench interface. A new query window is open with the SQL command `INSERT INTO studentstests(student_id, test_id) VALUES(1, 8);`. Below the query, the 'Output' tab displays the log entry:

#	Time	Action
1	08:47:51	INSERT INTO studentstests(student_id, test_id) VALUES(1, 8)

Рисунок 4.3 – Реєстрація студента на курс

Тепер змоделюємо проходження тесту певним студентом.

Студент з ідентифікатором 3 здаватиме тест з ідентифікатором 2.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						35

The screenshot shows a MySQL Workbench interface with a query editor at the top containing the SQL command: `1 • SELECT * FROM studentstests`. Below the editor is a "Result Grid" table with four columns: `students_test_id`, `student_id`, `test_id`, and `completed_date`. The data in the grid is as follows:

	<code>students_test_id</code>	<code>student_id</code>	<code>test_id</code>	<code>completed_date</code>
1	1	10	NULL	
2	1	11	NULL	
4	3	1	NULL	
5	3	2	NULL	
6	3	3	NULL	
7	6	4	NULL	
8	6	5	NULL	
9	6	6	NULL	
10	12	7	NULL	
11	12	8	NULL	
12	12	9	NULL	
13	1	7	NULL	
14	1	8	NULL	
*	NULL	NULL	NULL	NULL

Рисунок 4.4 – Вибірка з списку реєстрацій на тести

Для цього йому потрібно дати відповіді на два запитання.

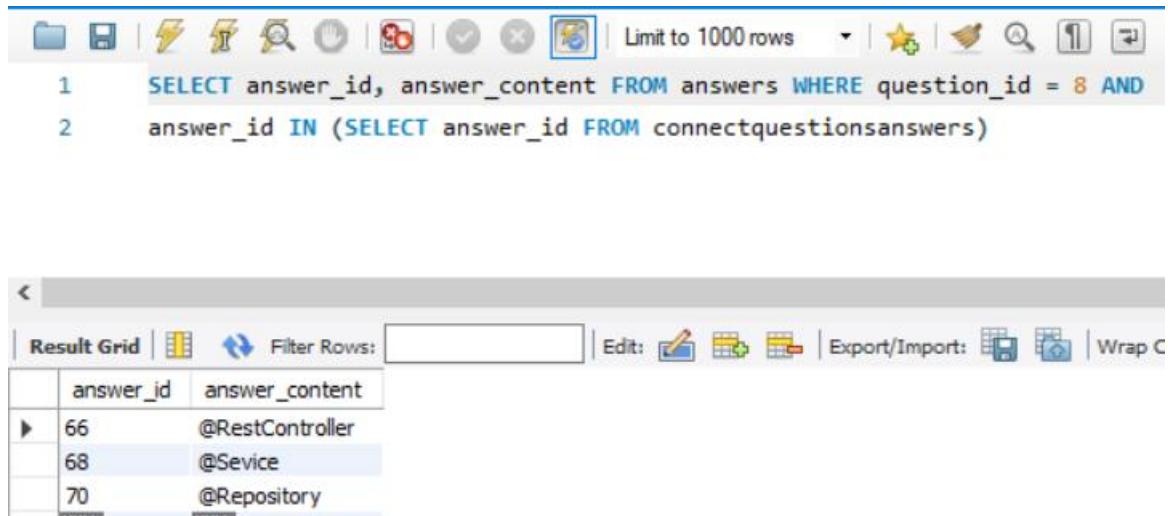
The screenshot shows a MySQL Workbench interface with a query editor at the top containing the SQL command: `1 • SELECT * FROM questions WHERE test_id = 2`. Below the editor is a "Result Grid" table with four columns: `question_id`, `test_id`, `content`, and `question_type`. The data in the grid is as follows:

	<code>question_id</code>	<code>test_id</code>	<code>content</code>	<code>question_type</code>
▶	2	2	What SQL stands for?	OPEN
▶	3	2	What CRUD stands for?	OPEN
*	NULL	NULL	NULL	NULL

Рисунок 4.5 – Запитання з тесту 2

Також виберемо список відповідей на це запитання, для цього виконаємо наступні два запити (для лівої та правої колонки відповідей).

Ми знаємо номер запитання – 8. Отже для підвищення продуктивності ми не повторяємо вибірки для цього запита (аплікація, яка використовуватиме цю базу повинна зберегти question_id з попереднього запиту та використати його для побудови наступних).



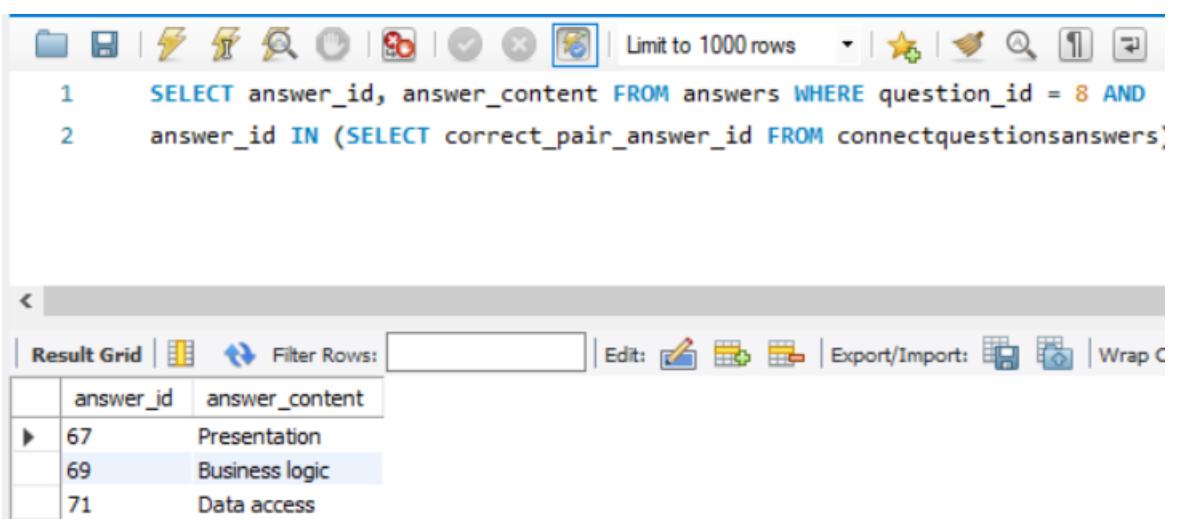
The screenshot shows the MySQL Workbench interface with two queries in the SQL editor:

```
1  SELECT answer_id, answer_content FROM answers WHERE question_id = 8 AND
2  answer_id IN (SELECT answer_id FROM connectquestionsanswers)
```

The result grid displays the following data:

answer_id	answer_content
66	@RestController
68	@Service
70	@Repository

Рисунок 4.9 – Вибір лівого стовпчика відповідей



The screenshot shows the MySQL Workbench interface with two queries in the SQL editor:

```
1  SELECT answer_id, answer_content FROM answers WHERE question_id = 8 AND
2  answer_id IN (SELECT correct_pair_answer_id FROM connectquestionsanswers)
```

The result grid displays the following data:

answer_id	answer_content
67	Presentation
69	Business logic
71	Data access

Рисунок 4.9 – Вибір лівого стовпчика відповідей

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					38

Тепер проведемо збереження відповіді студента до бази даних.

The screenshot shows the MySQL Workbench interface. In the SQL tab, three INSERT statements are run:

```
1 • INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 66, 67, NOW());
2 • INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 68, 71, NOW());
3 • INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 70, 69, NOW());
```

In the Output tab, the results of these insertions are displayed in a table:

#	Time	Action	Message
1	20:27:34	INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 66, 67, NOW())	1 row(s) affected
2	20:27:34	INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 68, 71, NOW())	1 row(s) affected
3	20:27:34	INSERT INTO studentconnectquestionsanswers(student_id, answer_id, pair_answer_id, submitted_time) VALUES(12, 70, 69, NOW())	1 row(s) affected

Рисунок 4.10 – Збереження відповіді студента на завдання типу поєднання

Продемонструємо процес збереження відповіді студента на запитання тестового типу.

Виберемо завдання такого типу.

The screenshot shows the MySQL Workbench interface. In the SQL tab, a query is run to select all rows from the questions table where the question_type is 'TEST':

```
5 • SELECT * FROM questions WHERE question_type = 'TEST'
```

The result grid displays the following data:

question_id	test_id	content	question_type
1	1	Where databases store data?	TEST
4	3	Check existing relations types	TEST
6	5	What JDBC stands for?	TEST
12	11	What is Kafka	TEST

Рисунок 4.11 – Вибір завдань тестового типу.

Зупинемось на запитання з ідентифікатором 1.

Оберемо студента для здачі цього завдання.

The screenshot shows the MySQL Workbench interface. In the SQL tab, two queries are run to find a student who can take a specific test:

```
1 • SELECT * FROM students WHERE student_id IN
2     (SELECT student_id FROM studentstests WHERE test_id IN
3         (SELECT test_id FROM questions WHERE question_id = 1))
```

The result grid displays the following data:

student_id	name	surname	group_id
3	John	Brain	NULL
NULL	NULL	NULL	NULL

4.12 – Вибір студента для здачу курсу

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						39

Тепер отримаємо список всіх варіантів відповідей на це питання.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query editor window displays the following SQL code:

```
1 •  SELECT answer_id, answer_content, is_right FROM answers WHERE question_id = 1
```

Below the code is a results grid titled "Result Grid". It has three columns: "answer_id", "answer_content", and "is_right". The data shows two rows:

	answer_id	answer_content	is_right
▶	51	Cache	1
	52	Hard drive	1

4.13 – Отримання варіантів відповідей на запитання

Відмітимо, що в даному випадку є більше однієї правильно відповіді на запитання, отже на стороні юзера інтерфейсу це питання буде відображене як питання з множинним вибором.

Збережемо відповідь студента.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query editor window displays the following SQL code:

```
1 •  INSERT INTO studenttestanswers(student_id, answer_id, submitted_time) VALUES(3, 51, NOW());
```

Below the code is an "Output" pane. It shows the action output table:

#	Time	Action
1	20:59:48	INSERT INTO studenttestanswers(student_id, answer_id, submitted_time) VALUES(3, 51, NOW())

To the right of the table, there's a "Message" section with the text "1 row(s) affected".

Рисунок 4.14 – Збереження відповіді студента на питання з множинним вибором

Для демонстрації збереження відповіді на питання відкритого типу використаємо питання з ідентифікатором 9

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						40

1 • SELECT * FROM questions WHERE question_type = 'TRUE_FALSE'

Result Grid

	question_id	test_id	content	question
▶	9	8	Should layered architecture always have 3 tiers?	TRUE...
	10	9	Do DropWizard always use Jetty as embedded server?	TRUE...
*	NULL	NULL	NULL	NULL

Рисунок 4.15 – Вибір запитання відкритого типу

2 (SELECT student_id FROM studentstests WHERE test_id IN
3 (SELECT test_id FROM questions WHERE question_id = 9))

Result Grid

	student_id	name	surname	group_id
▶	12	Dale	Smith	NULL
	1	Ihor	Polataiko	NULL
*	NULL	NULL	NULL	NULL

Рисунок 4.16 – Вибір студента для здачі питання

Збережемо відповідь студента з ідентифікатором 1.

1 INSERT INTO studenttruefalseanswers(student_id, question_id, answer, submitted_time) VALUES(1, 9, true, NOW());

Action Output

#	Time	Action	Message
1	21:37:38	INSERT INTO studenttruefalseanswers(student_id, question_id, answer, submitted_time) VALUES(1, 9, true, NOW())	1 row(s) affected

Рисунок 4.17 – Збереження відповіді на запитання типу true-false

Тепер перевіримо чи правильними були відповіді наших студентів, для цього потрібно пройтися по всіх запитаннях тесту та, в залежності від виду запитання

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						41

зробити вибірку з відповіді студента та правильної відповіді у відповідних таблицях нашої бази даних.

4.2.4 Вибірки даних – SELECT

В цьому розділі продемонструємо основні типи запитів для вибірки даних, які буде необхідно використовувати для користування базою даних.

Деякі дані наявні в цих вибірках були додані на етапі демонстрації роботи INSERT-запитів, проте база містить більшу кількість даних, ніж кількість тих даних, додавання яких до нашої бази було продемонстровано у даній курсовій роботі.

Дамп бази даних можна знайти, перейшовши за посиланням у додатку А.

```
SELECT students.student_id, name, surname FROM students
JOIN studentscourses
ON students.student_id = studentscourses.student_id
WHERE course_id = 7
```

student_id	name	surname
4	Martin	Fowler
5	Mary	Williams
6	Kate	Devis
7	Ann	Anderson
11	Alice	Liddell
12	Dale	Smith

Рисунок 4.18 – Вибірка всіх студентів, записаних на курс (ід 7)

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						42

```

SELECT courses.course_id, name FROM courses
JOIN studentscourses
ON courses.course_id = studentscourses.course_id
WHERE studentscourses.student_id = 8

```

course_id	name
5	DB basics
8	Advanced Java for web
6	Java advanced

Рисунок 4.19 – Вибірка всіх курсів, на які зареєстрований певний студент
(ід 8)

```

SELECT students.student_id, name, surname FROM students
JOIN studentstests
ON students.student_id = studentstests.student_id
WHERE test_id = 3

```

student_id	name	surname
3	John	Brain
2	Ed	Miller
8	Antony	Joshua

Рисунок 4.20 – Вибірка всіх студентів, які повинні здати певний тест
(ід 3)

```
SELECT test_id, topic_title FROM tests
WHERE course_id = 7
```

test_id	topic_title
7	Spring MVC essentials
8	Web architecture
9	DropWizard

Рисунок 4.21 – Вибірка всіх тестів з курсу (ід 7)

```
SELECT tests.test_id, topic_title FROM tests
JOIN studentstests
ON tests.test_id = studentstests.test_id
WHERE student_id = 8
AND (SELECT COUNT(*) FROM testattempts
WHERE testattempts.students_test_id = studentstests.students_test_id) = 0
```

test_id	topic_title
1	What is db?
2	Query language
3	Relations
10	gRPC, HTTP/2
11	Apache Kafka
4	Connection to sockets
5	Connection to DB
6	JMX tools

Рисунок 4.22 – Вибірка всіх незданих тестів для студента з ід 8

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					44

```
SELECT question_id, content, question_type FROM questions WHERE test_id = 2
```

question_id	content	question_type
2	What SQL stands for?	OPEN
3	What CRUD stands for?	OPEN
13	INSERT command belongs to	TEST
14	Connect commands of one type	CONNECT

Рисунок 4.23 – Вибірка всіх запитань з тесту з ід 2

```
SELECT answer_id, answer_content, is_right FROM answers WHERE question_id = 13
```

answer_id	answer_content	is_right
78	DDL	0
79	DML	1
80	DCL	0
81	DQL	0

Рисунок 4.24 – Вибірка всіх варіантів відповідей на тестове запитання (ід 13)

```
SELECT answer_id, answer_content FROM answers WHERE answer_id IN  
(SELECT answer_id FROM connectquestionsanswers WHERE question_id = 14)
```

answer_id	answer_content
82	INSERT
84	COMMIT
86	CREATE

Рисунок 4.25 – Вибірка лівосторонніх варіантів відповідей на запитання типу поєднання (ід 14)

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
45						

```

SELECT answer_id, answer_content FROM answers WHERE answer_id IN
(SELECT correct_pair_answer_id FROM connectquestionsanswers WHERE question_id = 14)

```

answer_id	answer_content
83	UPDATE
85	ROLL BACK
87	ALTER

Рисунок 4.26 – Вибірка правосторонніх варіантів відповідей на запитання типу поєднання (ід 14)

```

SELECT student_id, studentopenquestionanswers.question_id, answer AS student_answer,
answer_content AS correct_answer
FROM studentopenquestionanswers
JOIN answers ON studentopenquestionanswers.question_id = answers.question_id
WHERE studentopenquestionanswers.question_id = 2 AND student_id = 3

```

student_id	question_id	student_answer	correct_answer
3	2	Structured Query Language	Structured Query Language

Рисунок 4.27 – Перевірка відповіді студента з ід 3 на запитання відкритого типу (ід 2)

```

SELECT student_id, question_id, answer_id AS student_answer FROM studenttestanswers
WHERE question_id = 13 AND student_id = 8

```

student_id	question_id	answer_id
8	13	81
8	13	79

Рисунок 4.28 – Вибірка відповіді студента з ід 8 на запитання тестового типу (ід 13)

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					46

```

SELECT question_id, answer_id AS right_answer FROM answers
WHERE question_id = 13 AND answers.is_right = true

```

question_id	right_answer
13	79
13	80

Рисунок 4.29 – Вибірка правильної відповіді на запитання тестового типу (ід 13)

```

SELECT student_id, connectquestionsanswers.question_id, connectquestionsanswers.answer_id,
studentconnectquestionsanswers.pair_answer_id AS student_pair_choice,
connectquestionsanswers.correct_pair_answer_id AS correct_pair
FROM connectquestionsanswers
JOIN studentconnectquestionsanswers
ON studentconnectquestionsanswers.answer_id = connectquestionsanswers.answer_id
WHERE connectquestionsanswers.question_id = 14 AND student_id = 8

```

student_id	question_id	answer_id	student_pair_choice	correct_pair_answer_id
8	14	82	83	83
8	14	84	87	85
8	14	86	85	87

Рисунок 4.30 – Перевірка відповіді студента з ід 8 на запитання типу поєднання з ід 14

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						47

```

SELECT student_id, studenttruefalseanswers.question_id,
CASE WHEN studenttruefalseanswers.answer = 1 THEN
    "TRUE"
ELSE
    "FALSE"
END AS student_answer,
answers.answer_content AS correct_answer FROM studenttruefalseanswers
JOIN answers ON studenttruefalseanswers.question_id = answers.question_id
WHERE studenttruefalseanswers.question_id = 9 AND student_id = 1

```

student_id	question_id	student_answer	correct_answer
1	9	TRUE	TRUE

Рисунок 4.31 – Перевірка відповіді студента з ід 1 на запитання типу «так-ні» (ід 9)

```

SELECT student_id, studenttruefalseanswers.question_id,
CASE WHEN studenttruefalseanswers.answer = 1 THEN
    "TRUE"
ELSE
    "FALSE"
END AS student_answer,
answers.answer_content AS correct_answer, submitted_time FROM studenttruefalseanswers
JOIN answers ON studenttruefalseanswers.question_id = answers.question_id
WHERE studenttruefalseanswers.question_id = 10 AND student_id = 5
ORDER BY submitted_time ASC

```

student_id	question_id	student_answer	correct_answer	submitted_time
5	10	FALSE	TRUE	2019-12-18 16:28:54.000000
5	10	TRUE	TRUE	2019-12-19 16:27:51.000000
5	10	FALSE	TRUE	2019-12-20 16:28:43.000000

Рисунок 4.32 – Відображення кількох відповідей студента (ід 5) на запитання (ід 10)

Ситуація показана на рисунку 4.32 виникає тоді, коли студент використав кілька спроб здача одного тесту. В такому випадку ми можемо обрати яку відповідь йому зараховувати.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						48

```

SELECT student_id, test_id, COUNT(*) AS number_of_attempts FROM testattempts
JOIN studentstests
ON testattempts.students_test_id = studentstests.students_test_id
WHERE student_id = 8 AND test_id = 2

```

student_id	test_id	number_of_attempts
8	2	1

Рисунок 4.33 – Відображення кількості використаних студентом (ід 8) спроб здати тест (ід 2)

4.2.3 Оновлення даних – UPDATE

В цій частині продемонструємо роботу основних запитів на оновлення даних у базі даних «Тестування знань»

В процесі тестування бази даних було виявлено, що одну з груп студентів названо неправильно.

```

1   SELECT group_name FROM `groups` WHERE group_id = 5

```

group_name
lifetime-learners

Рисунок 4.34 – Вибірка групи названої з граматичною помилкою

Виправимо цю помилку виконавши наступну команду

```

1   UPDATE `groups` SET group_name = 'lifetime-learners' WHERE group_id = 5

```

#	Time	Action
1	15:40:29	UPDATE 'groups' SET group_name = 'lifetime-learners' WHERE group_id = 5

Рисунок 4.35 – Оновлення назви групи

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						49

Тепер додамо модератора до цієї групи, для цього встановимо значення 2 зовнішнього ключа moderator_id для запису з таблиці groups з group_id 4.

The screenshot shows the MySQL Workbench interface. In the SQL tab, there is a single query: `1 UPDATE `groups` SET moderator_id = 2 WHERE group_id = 4`. Below it, the Output tab shows the result of the query: `1 15:52:45 UPDATE `groups` SET moderator_id = 2 WHERE group_id = 4`. The status is indicated by a green checkmark.

Рисунок 4.36 – Додавання модератора до групи

Тепер додамо кількох студентів (ід: 1, 2, 7, 11) до групи з ід 4.
(Відповідно до дизайну бази студент може належати максимум до однієї групи).

The screenshot shows the MySQL Workbench interface. In the SQL tab, there is a single query: `1 UPDATE students SET group_id = 4 WHERE student_id IN (1, 2, 7, 11)`. Below it, the Output tab shows the result of the query: `1 16:03:59 UPDATE students SET group_id = 4 WHERE student_id IN (1, 2, 7, 11)`. The status is indicated by a green checkmark.

Рисунок 4.37 – Додавання студентів до групи

Перевіримо результати додавання студентів до групи.

The screenshot shows the MySQL Workbench interface. In the SQL tab, there is a single query: `1 SELECT * FROM students`. Below it, the Results tab displays a table titled "Result Grid". The table has columns: student_id, name, surname, and group_id. The data is as follows:

	student_id	name	surname	group_id
▶	1	Ihor	Polataiko	4
	2	Ed	Miller	4
	3	John	Brain	NULL
	4	Martin	Fowler	NULL
	5	Mary	Williams	NULL
	6	Kate	Devis	NULL
	7	Ann	Anderson	4
	8	Antony	Joshua	NULL
	9	Kate	Sierra	NULL
	10	Alan	Brown	NULL
	11	Alice	Liddell	4
	12	Dale	Smith	NULL

Рисунок 4.38 – Вибірка всіх студентів

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						50

4.2.4 Видалення даних – DELETE

В цій частині продемонструємо роботу основних запитів на видалення даних у базі даних «Тестування знань»

Видалення даних з бази не є рекомендованою практикою, оскільки може привести до втрати цінних даних. В робочих моделях частіше користуються деактивацією, тобто вводять додаткове поле active типу boolean до таблиці і при потребі видалити дані значення цього поля змінюють на false. В такому випадку дані фізично не видаляються з бази даних, що знижує ризик втрати важливої інформації, проте в той самий час ми можемо умовно «деактивувати» їх.

Для прикладу продемонструємо видалення студента з ід 12 з нашої системи.

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL query:

```
1      DELETE FROM students WHERE student_id = 12
```

In the bottom-right pane, there is an "Output" window titled "Action Output". It contains a table with one row:

#	Time	Action
1	16:19:28	DELETE FROM students WHERE student_id = 12

The row has a red error icon next to it, indicating that the operation failed.

Рисунок 4.39 – Спроба видалити студента з ід 12 з бази даних

Як бачимо, в результаті виконання такого запиту отримаємо помилку, що пов'язано з тим, що запис з таблиці students з student_id 12 є зовнішнім в інших таблицях і при видаленні такого запису можливі три варіанти:

- виникне помилка при видаленні такого запису
- всі значення зовнішніх ключів, які вказують на даний запис, будуть оновлені значенням NULL
- всі записи, які містять ідентифікатор даного запису як зовнішній ключ будуть видалені (cascade)
- видалення відбудеться, але зв'язок буде зламано

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						51

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a SQL editor window containing the following query:

```
1   SELECT * FROM studentstests WHERE student_id = 12
```

Below the SQL editor is a "Result Grid" window. It has a header row with columns: students_test_id, student_id, test_id, and max_attempts_allowed. The data rows are:

	students_test_id	student_id	test_id	max_attempts_allowed
▶	10	12	7	3
	11	12	8	3
	12	12	9	3
	45	12	10	3
	46	12	11	3

Рисунок 4.40 – Записи, для яких студент з ід 12 є зовнішнім ключем

Для того, щоб даної помилки не виникало, виконаємо наступну команду, що дозволить нам видалити цього студента з нашої бази даних.

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a SQL editor window containing the following command:

```
1   set foreign_key_checks = 0
```

Below the SQL editor is an "Output" window. It shows the following log entry:

#	Time	Action
1	16:42:54	set foreign_key_checks = 0

Рисунок 4.41 – Вимкнення перевірки зовнішніх ключів

Тепер повторимо виконання команди на видалення запису з таблиці students.

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a SQL editor window containing the following command:

```
1   DELETE FROM students WHERE student_id = 12
```

Below the SQL editor is an "Output" window. It shows the following log entry:

#	Time	Action
1	16:47:32	DELETE FROM students WHERE student_id = 12

Рисунок 4.42 – Видалення студента з ід 12 з бази даних

Як бачимо з рисунку 4.42, тепер спроба видалення пройшла успішно.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						52

Тепер перевіримо що сталося з записами з таблиці StudentsTests, що мали зовнішні ключі, які посилалися на запис, який ми щойно видалили.

```
1   SELECT * FROM studentstests WHERE students_test_id IN (10, 11, 12, 45, 46)
```

students_test_id	student_id	test_id	max_attempts_allowed
10	12	7	3
11	12	8	3
12	12	9	3
45	12	10	3
46	12	11	3

Рисунок 4.43 – Вибірка тестів студент з ід 12 з таблиці StudentsTests

В даному випадку відбувся четвертий з можливих варіантів, тобто зовнішні ключі збережено, але зв'язок втрачено. Це не рекомендована практика.

Щоб зберегти забезпечити каскадне видалення даних необхідно встановити наступну умову на зовнішній ключ

```
ON DELETE CASCADE
```

Щоб зберегти забезпечити зміну значень зовнішніх ключів на NULL при видаленні запису, на який вони посилаються, необхідно встановити наступну умову на зовнішній ключ :

```
ON DELETE SET NULL
```

Щоб забезпечити відсутність дії, необхідно встановити наступну умову на зовнішній ключ :

```
ON DELETE NO ACTION
```

Щоб зберегти забезпечити зміну значень зовнішніх ключів на значення по замовчуванню при видаленні запису, на який вони посилаються:

```
ON DELETE SET DEFAULT
```

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
						53

ВИСНОВКИ

Задачею даної курсової роботи було створення бази даних «Тестування знань».

В процесі роботи на даним завданням було:

- проаналізовано задачу та проблемну область
- розроблено дизайн бази даних у вигляді ER діаграми
- створено та наповнено початковими даними базу
- продемонстровано виконання базових запитів у даній базі даних

Результатом даної роботи є MVP (Minimum Value Product) модель бази даних, яка, попри те, уже ж готовою до застосування при побудові систем тестування знань.

Під час роботи над даною курсовою роботою було покращено знання про SQL бази даних, а також набуто досвіду розробки дизайну реляційних баз даних.

Зм.	Арк.	№ докум.	Підпис	Дата	Арк.
					KP.IПЗ-03.П3

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Learning SQL / O'Reilly Media: Alan Beaulieu.
2. SQL Cookbook / Anthony Molinaro.
3. Head First SQL: Your Brain on SQL - A Learner's Guide / Lynn Beighley.
4. Relational Database Design and Implementation: Clearly Explained / Jan L. Harrington.
5. SQL | DDL, DQL, DML, DCL and TCL Commands // URL:
<https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands>.
6. SQL (Wikipedia) // URL: <https://en.wikipedia.org/wiki/SQL>
7. NoSQL // URL: <https://www.ibm.com/cloud/learn/nosql-databases>
8. SQL // URL: <https://www.khanacademy.org/computing/computer-programming/sql>
9. Beginning database design / Clare Churcher.
10. Database Design and Relational Theory: Normal Forms and All That Jazz / Christopher J. Date.

Зм.	Арк.	№ докум.	Підпис	Дата	КП.ІПЗ-03.ПЗ	Арк.
55						

ДОДАТОК А

Проект на [github.com](#)

Посилання на проект на [github.com](#):

<https://github.com/Igor-Polatajko/KnowledgeTestingSystemDB>