МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка бази даних для формування білетів з фахового іспиту

Виконав студент групи **К-19б**
спеціальності 122 Комп'ютерні науки
Захаркін Альберт Вікторович

Керівник **асистент**
Клепатська В.В.

Консультант **д.т.н., професор**
Казакова Н.Ф.

Рецензент **канд.техн.наук., доцент**
Перелигін Б.В.

Одеса 2023

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

База даних (БД) – це організована колекція даних, що зберігаються та оброблюються за допомогою комп'ютерної системи.

Таблиці – це структуровані сутності, що складаються з рядків (записів) та стовпців (полів).

Схема – визначає структуру таблиць та його відносини друг з одним та включає опис таблиць, полів, обмежень цілісності та зв'язків між таблицями.

Запити – використовуються для отримання та оновлення даних у базі даних.

Індекси – це спеціальні структури даних, створені для прискорення пошуку та доступу до даних.

Обмеження цілісності – правила та умови, визначені для забезпечення цілісності даних.

Транзакції – угруповання операцій бази даних до логічно пов'язаних одиниць роботи.

ACID-властивості – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

Система управління базами даних – набір взаємопов'язаних даних і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Первинний ключ (Primary Key) – унікальний ідентифікатором для кожного запису в таблиці, складається з одного або декількох стовпців таблиці.

Зовнішні ключі (Foreign Key) – встановлюють зв'язки між таблицями, вони посилаються на значення первинного ключа в іншій таблиці, створюючи зв'язок між ними.

Дані – це фактична інформація, яка може бути збережена, оброблена та використана для різних цілей, вони можуть бути представлені у різних форматах, таких як текст, числа, зображення, відео, аудіо тощо.

Запити до бази даних (БД) - це команди або запити, які виконуються до БД, щоб отримати певну інформацію або здійснити дії з даними; запити дозволяють вибирати, вставляти, оновлювати або видаляти дані з БД, а також здійснювати різні операції над даними.

БД – база даних.

СУБД – системи управління базами даних, що використовується.

SQL – Structured Query Language.

OLAP – Online Analytical Processing.

PK – Primary Key

FK – Foreign Key

ВСТУП

Створення бази даних залишається актуальним та важливим завданням у сучасному світі з наростаючим обсягом даних та потребою в їх управлінні.

База даних (БД) – це організована колекція даних, що зберігаються та оброблюються за допомогою комп'ютерної системи. Вона призначена для ефективного зберігання, керування та вилучення даних. БД складається із структурованих даних, які організовані в таблиці або інші структури даних, такі як дерева, графи та документи, залежно від системи управління базами даних, що використовується (СУБД).

Мета кваліфікаційної роботи бакалавра полягає у створенні структурованого і ефективного засобу зберігання та керування інформацією, необхідною для формування білетів з іспиту з певної фахової галузі. Основна мета полягає у забезпеченні легкого доступу до запитів, які дозволять згенерувати білети з відповідними питаннями та відповідями.

Нижче представлено кілька аргументів, які підтверджують актуальність створення баз даних.

В першу чергу це пов’язано зі збільшення обсягу даних. Обсяги даних постійно зростають, особливо у контексті цифрової економіки та збір інформації з різних джерел. База даних допомагає ефективно зберігати, індексувати та керувати цими великими обсягами даних.

Якщо мова йде про швидкість та продуктивність, база даних дозволяє оптимізувати роботу з даними та виконання запитів, забезпечуючи швидкий доступ до інформації. Це особливо важливо, коли необхідно обробляти великі обсяги даних у реальному часі.

База даних дозволяє встановлювати правила цілісності та обмеження для забезпечення правильності та консистентності даних. Це важливо для запобігання помилок, дублювання даних та забезпечення їхньої надійності.

Спільний доступ до база даних дозволяє кільком користувачам або програмам одночасно працювати з даними, що забезпечує контроль доступу

та управління правами користувачів. Це в свою чергу дозволяє ефективно співпрацювати та обмінюватись даними в організації чи команді.

Забезпечення безпеки даних теж є одним з актуальних моментів. База даних дозволяє встановлювати заходи безпеки, такі як шифрування, автентифікація та контроль доступу, для захисту важливих даних від несанкціонованого доступу чи втрати [1].

З точки аналітика та прийняття рішень, база даних надає зручну основу для аналізу даних, створення звітів та прийняття обґрунтованих рішень на основі доступної інформації.

Також, дозволяє скоротити збереження фізичних документів, тобто замість зберігання великої кількості фізичних документів, база даних дозволяє електронно зберігати та керувати даними, що сприяє економії місця та полегшує їх пошук та оновлення.

Все більше сфер діяльності вимагають ефективного управління даними, що робить створення бази даних актуальним та важливим процесом для багатьох організацій та підприємств.

Дана кваліфікаційна робота бакалавра, складається з 53 сторінок, 1 таблиці 14 рисунків та 5 джерел посилання.

1 ХАРАКТЕРИСТИКА ОСНОВНИХ ЕЛЕМЕНТІВ БАЗИ ДАНИХ

Створення бази даних може бути необхідним у багатьох випадках , коли потрібно ефективне зберігання , керування та доступ до даних . Ось кілька прикладів, коли створення БД може бути корисним.

При розробці веб-застосувань, БД може використовуватись для зберігання користувальницьких даних , інформації про продукти, замовлення, коментарі та інші сутності , необхідні для роботи додатки . Якщо мова йде про бізнес, при управлінні клієнтськими даними, тобто при необхідності керувати інформацією про клієнтів, їх контактних даних, замовленнях, історії взаємодій та інше, БД може допомогти зберігати ці дані структуровано та забезпечувати зручний доступ до них. В освітніх установах або платформах для онлайн-навчання БД може використовуватись для зберігання інформації про курси, учнів, прогрес навчання, завдання та оцінки, тобто у системах управління навчальними матеріалами. Для компаній, що займаються продажами товарів та управлінням запасами, БД може бути корисна для відстеження складських запасів, управлінням поставками, відстеженням продажів та обліку товарів. Створення БД може допомогти збирати та аналізувати дані для отримання бізнес-аналітики, створення звітів та прийняття поінформованих рішень. У медичній сфері БД використовуються для зберігання медичних записів пацієнтів, результатів аналізів, призначень та інших даних, необхідних для управління охороною здоров'я.

Дані можуть бути структурованими або неструктурованими. Структуровані дані відповідають певній організації або схемі, яка визначає, як дані пов'язані між собою. Прикладами структурованих даних є таблиці у базі даних, де кожен стовпець представляє певний тип даних, а кожен рядок містить значення для кожного стовпця.

Неструктуровані дані не мають жорсткої організації або схеми. Вони можуть включати, наприклад, тексти, документи, електронну пошту, вебсторінки, соціальні медіа повідомлення тощо. Обробка неструктурованих да-

них може бути складнішою, оскільки потрібно розуміти контекст та структуру даних для витягування корисної інформації.

Дані є цінним ресурсом, який використовується для прийняття рішень, аналізу, виявлення тенденцій, створення звітів та багато інших завдань. Зберігання та управління даними відбувається за допомогою баз даних, які забезпечують структуровану і організовану систему для зберігання та доступу до даних.

Загалом БД корисна у будь-якій ситуації, коли потрібне ефективне зберігання, організація та доступ до даних, а також підтримка операцій, які пов'язані з цими даними.

1.1 Опис основних компонентів бази даних

Основні компоненти бази даних включають (див.рис.1) таблиці, схеми, запити, індекси, обмеження цілісності, транзакції.

«Таблиці» – це структуровані сутності, що складаються з рядків (записів) та стовпців (полів). Кожен рядок представляє окремий запис, а кожен стовпець містить певний тип даних. «Схема», визначає структуру таблиць та його відносини друг з одним. Включає опис таблиць, полів, обмежень цілісності та зв'язків між таблицями. «Запити» – використовуються для отримання та оновлення даних у базі даних. Запити можуть виконуватися для вибірки певних даних, виконання обчислень, об'єднання даних із різних таблиць та інших операцій. «Індекси» – це спеціальні структури даних, створені для прискорення пошуку та доступу до даних. Індекси зазвичай створюються на ключових полях таблиці. «Обмеження цілісності» – правила та умови, визначені для забезпечення цілісності даних. Вони можуть включати обмеження на унікальність значень, цілісність посилань, обмеження перевірки та інші. «Транзакції», угруповання операцій бази даних до логічно пов'язаних одиниць роботи. Транзакції забезпечують атомарність, узгодженість, ізольованість та стійкість даних (ACID-властивості). Бази даних використовуються в

різних галузях, таких як бізнес, наукові дослідження, інтернет-додатки, управління контентом, фінанси та інші, для ефективного зберігання та управління даними, забезпечення безпеки та обробки запитів.

Рисунок 1 – Основні компоненти БД

База даних складається з різних елементів, які спільно забезпечують зберігання, організацію та обробку даних. До основними елементів бази даних також можна віднести:

– «Таблиці» (таблиці є основною одиницею організації даних в базі даних; вони складаються з рядків і стовпців і використовуються для зберігання конкретного типу інформації);
– «Стовпці» (стовпці представляють окремі поля або атрибути даних у таблиці; вони визначають тип даних, який може бути збережений у відповідному стовпці);

- «Рядки» (рядки в таблицях містять фактичні дані; кожен рядок відповідає запису або кортежу, що містить значення для кожного стовпця в таблиці);
- «Ключі» (ключі використовуються для унікальної ідентифікації записів в таблицях; основні ключі гарантують унікальність значень, а зовнішні ключі встановлюють зв'язки між таблицями);
- «Запити» (запити використовуються для отримання, оновлення або видалення даних з бази даних; вони дозволяють вибирати певні рядки або комбінації рядків на основі заданих умов);
- «Індекси» (індекси використовуються для прискорення пошуку та доступу до даних; вони створюються на одному або кількох стовпцях таблиці, що дозволяє ефективно виконувати запити, які використовують ці стовпці в умовах пошуку);
- «Відношення між таблицями» (відношення або зв'язки встановлюються між таблицями, що дозволяють створювати зв'язані дані; зв'язки можуть бути один до одного, один до багатьох або багато до багатьох);
- «Тригери» (тригери використовуються для визначення автоматичних дій або обробки, які виконуються автоматично при певних подіях, таких як вставка, оновлення або видалення даних);
- «Процедури та функції» (процедури та функції дозволяють зберігати блоки коду, які можуть бути викликані для виконання певних операцій або обчислень).

1.1.1 Опис компонента БД – «Таблиці»

Таблиці є основними структурними компонентами [2] бази даних і використовуються для зберігання і організації даних у табличній формі. Кожна таблиця містить набір стовпців і рядків, які представляють атрибути і записи

відповідно. Нижче представлено кілька важливих аспектів, пов'язаних з таблицями в базі даних:

– назва таблиці;
– структура таблиці;
– первинний ключ;
– зовнішні ключі;
– записи;
– запити до таблиць;
– індекси.

Кожна таблиця має унікальне ім'я, що ідентифікує її всередині бази даних – назву таблиці.

Структура таблиці визначає список стовпців, які вона містить, а також їх типи даних. Кожен стовпець має ім'я, тип даних (наприклад, ціле число, рядок, дата тощо) і, за потреби, додаткові обмеження, такі як обов'язковість заповнення (NOT NULL) або унікальність значень (UNIQUE).

Первинний ключ (Primary Key) є унікальним ідентифікатором для кожного запису в таблиці. Він може складатися з одного або декількох стовпців таблиці. Первинний ключ забезпечує унікальність записів і дозволяє швидкий доступ до конкретних записів.

Зовнішні ключі (Foreign Key) встановлюють зв'язки між таблицями. Вони посилаються на значення первинного ключа в іншій таблиці, створюючи зв'язок між ними. Зовнішні ключі використовуються для побудови зв’язків і забезпечення цілісності даних.

Кожен рядок таблиці представляє окремий запис (rows) або кортеж даних. Кожен стовпець таблиці містить значення для конкретного атрибуту цього запису.

Для отримання, оновлення або видалення даних з таблиці використовуються запити SQL (Structured Query Language). Запити дозволяють вибрати конкретні записи, виконати обчислення або змінити дані в таблиці.

Індекси використовуються для прискорення пошуку та доступу до даних в таблиці. Вони створюються на одному або кількох стовпцях і дозволяють швидкий доступ до записів, що задовольняють певні умови.

Таблиці є основними засобами організації та зберігання даних в базі даних і дозволяють ефективно виконувати операції з даними, такі як пошук, сортування, фільтрація та аналіз.

1.1.2 Опис компонента БД – «Запити»

Запити в базі даних використовуються для отримання, оновлення, видалення або вставки даних в таблицях бази даних. Запити використовуються для взаємодії з даними і виконання різних операцій з ними. Найпоширенішою мовою для написання запитів є мова структурованих запитів SQL. Основні типи запитів включають SELECT, UPDATE, INSERT, DELETE, JOIN і AGGREGATE.

Запити вибірки (SELECT) використовуються для отримання даних з бази даних. Можна вибрати конкретні стовпці для виведення, використовувати фільтри для вибору певних записів та використовувати умови для складніших запитів.

Запити оновлення (UPDATE) використовуються для зміни існуючих даних в базі даних. Запит дозволяє оновлювати значення в конкретних стовпцях для вибраних записів або застосовувати оновлення до всіх записів, що відповідають певним умовам.

Запити вставки (INSERT) використовуються для додавання нових записів до таблиці бази даних. З їхньою допомогою можна вказати значення для кожного стовпця нового запису або вставити дані з іншої таблиці.

Запити видалення (DELETE) використовуються для видалення записів з таблиці бази даних, можна видалити всі записи або видалити лише ті записи, що відповідають певним умовам.

Запити об'єднання (JOIN) використовуються для комбінування даних з двох або більше таблиць на основі спільних стовпців. Вони дозволяють отримувати дані зв'язаних таблиць у одному запиті.

Запити агрегації (AGGREGATE) використовуються для обчислення сум, середніх значень, максимальних і мінімальних значень та інших агрегатних функцій для груп записів. Вони в свою чергу дозволяють отримати статистичні дані про дані в базі даних.

Запити в базі даних дозволяють отримувати потрібні дані, змінювати їх і виконувати складні операції з даними. Вони є потужним інструментом для роботи з базою даних і виконання різноманітних завдань обробки даних.

1.1.3 Опис компонента БД – «Індекси»

Індекси в БД використовуються для покращення швидкості пошуку та фільтрації даних. Вони дозволяють прискорити виконання запитів до БД, зменшити кількість операцій пошуку та покращити продуктивність системи загалом. Індекси створюються на певних стовпцях таблиць і забезпечують структуроване упорядкування даних для швидшого доступу.

Основні переваги використання індексів в БД описані нижче.

Покращена продуктивність запитів, тобто індекси дозволяють швидше знаходити потрібні дані, оскільки вони створюють структуру, що спрощує процес пошуку. Запити, які використовують індексовані стовпці, виконуються швидше, що дозволяє отримувати результати запитів за короткий час.

Індекси допомагають зменшити кількість операцій пошуку, які потрібно виконати для отримання результатів запитів. Це зменшує навантаження на систему і підвищує її продуктивність.

За допомогою індексів можна швидко фільтрувати дані за певними умовами, оскільки індекси дозволяють швидко знаходити відповідні записи без необхідності сканування всієї таблиці.

Індекси можуть бути використані для встановлення унікальності значень в певному стовпці таблиці. Це дозволяє забезпечити цілісність даних та запобігти вставці дублікатів.

Також, індекси мають деякі недоліки, а саме:

- потреба в додатковому просторі (індекси займають додатковий простір в БД, оскільки вони зберігаються окремо від основних даних; це може вплинути на розмір БД і потребує додаткового місця для збереження);
- повільне оновлення даних (при внесенні змін до даних, які індексуються, потрібно оновити відповідні індекси; це може зайняти деякий час та вплинути на продуктивність системи під час оновлення великих обсягів даних);
- зайві індекси (неправильне використання індексів може призвести до створення зайвих індексів, які не ефективно використовуються або надмірно займають простір; це може сповільнити роботу БД та зайняти зайві ресурси).

Вибір використання індексів залежить від конкретних потреб проєкту та характеристик БД. Необхідно розглянути типи запитів, які використовуються, розмір даних, частоту оновлення даних та інші фактори для визначення оптимального використання індексів у конкретному випадку.

1.1.4 Опис компонента БД – «Відношення між таблицями»

В базі даних відношення між таблицями використовуються для встановлення зв'язків та залежностей між ними. Це дозволяє створювати складні структури даних та забезпечувати цілісність та консистентність інформації. Основні типи відношень між таблицями включають:

- один до одного (One-to-One);
- один до багатьох (One-to-Many);
- багато до одного (Many-to-One);

– багато до багатьох (Many-to-Many).

У типі відношення «Один до одного (One-to-One)» кожен запис в одній таблиці співставляється з одним записом в іншій таблиці. Цей тип відношення використовується, коли існує пряма однозначна залежність між двома сутностями.

У типі відношення «Один до багатьох (One-to-Many)» кожен запис в одній таблиці може мати багато відповідних записів в іншій таблиці. Наприклад, один автор може мати багато книжок. В такому випадку таблиця авторів матиме відношення один до багатьох з таблицею книжок.

«Багато до одного (Many-to-One)» – це зворотне відношення до відношення «Один до багатьох». В цьому випадку багато записів в одній таблиці посилаються на один запис в іншій таблиці. Наприклад, багато студентів можуть бути призначені до одного викладача.

В типі відношень «Багато до багатьох (Many-to-Many)» кожен запис в одній таблиці може мати багато відповідних записів в іншій таблиці, і навпаки. Це досягається за допомогою додаткової проміжної таблиці, яка містить зв'язки між записами двох таблиць. Наприклад, багато студентів можуть бути зареєстровані на багато курсів, і кожен студент може мати багато курсів.

Відношення між таблицями в базі даних дозволяють організувати комплексні структури даних та забезпечувати зв'язок між сутностями. Вони використовуються для забезпечення цілісності даних, забезпечення ефективного зберігання та отримання даних, а також для виконання операцій об'єднання, фільтрації та агрегації даних.

1.2 Аналіз основних аспектів безпеки БД

Безпека баз даних є критичним аспектом управління і збереження даних. Забезпечення безпеки баз даних важливо для запобігання несанкціонованому доступу, втрати чи пошкодженню даних, а також збереження конфіденційності, цілісності та доступності даних.

Основні аспекти безпеки баз даних включають:

– аутентифікація та авторизація;
– шифрування даних;
– запобігання вразливостям;
– резервне копіювання та відновлення;
– моніторинг та аудит безпеки;
– фізична безпека.

Аутентифікація та авторизація – це процес визначення та перевірки ідентифікації користувачів, а також надання прав доступу до різних об'єктів бази даних відповідно до їхніх ролей та повноважень. Забезпечення сильних паролів, двофакторної аутентифікації та строго контролю доступу є важливими елементами безпеки баз даних [3].

Захист даних шляхом шифрування є ефективним способом запобігання несанкціонованому доступу до конфіденційної інформації. Для цього можна використовувати шифрування на рівні бази даних, файлів, комунікаційного каналу та засоби шифрування даних на рівні додатків.

Безпека баз даних вимагає активного моніторингу та патчінгу системи, а також застосування кращих практик у процесі розробки та експлуатації бази даних. Варто враховувати потенційні вразливості, такі як SQL-ін'єкції, переповнення буфера, викрадення ідентифікаторів сеансу та інші, і приділяти належну увагу їхньому усуненню.

Важливо мати ефективні процедури резервного копіювання та відновлення даних. Регулярне створення резервних копій баз даних дозволяє відновити дані у разі випадкового видалення, пошкодження або втрати.

Систематичний моніторинг бази даних, аналіз журналів подій та ведення аудиту дозволяють виявляти та реагувати на потенційні загрози безпеці. Це включає в себе виявлення спроб несанкціонованого доступу, змін у даних або інших підозрілих активностей.

Крім технічних заходів безпеки, важливо забезпечити фізичну безпеку серверних приміщень, дата-центрів та засобів зберігання даних.

Вибір відповідних заходів безпеки баз даних залежить від конкретних вимог і контексту проєкту. Рекомендується ретельно проаналізувати потенційні загрози безпеці, дотримуватися кращих практик та консультуватися з експертами у сфері безпеки баз даних для вибору найкращих рішень відповідно до потреб.

1.3 Аналіз методів та інструменти для аналізу даних в БД

Для аналізу даних в базах даних існує різноманітні методи та інструменти, які допомагають виявляти тренди, залежності та отримувати цінну інформацію.

SQL (Structured Query Language) є стандартною мовою запитів, яка використовується для отримання даних з бази даних. Вона надає можливість формулювати складні запити для вибірки, сортування, групування та агрегації даних.

OLAP (Online Analytical Processing) є методологією аналізу даних, спрямованою на швидкий та багатоваріантний аналіз великих обсягів даних. Вона дозволяє виконувати складні аналітичні операції, такі як зведені таблиці, зрізи даних, багатовимірний аналіз та інші.

Data Mining використовується для виявлення та вивчення прихованих залежностей, трендів та моделей в базах даних. Цей підхід використовує різні алгоритми та методи для автоматичного виявлення цінної інформації з великих обсягів даних.

Data Visualization (візуалізація даних) використовується для графічного представлення даних, що допомагає в розумінні та сприйнятті великих обсягів інформації. Інструменти для візуалізації даних дозволяють створювати діаграми, графіки, теплові карти та інші візуальні елементи для подання даних у зрозумілій формі.

Statistical Analysis (статистичний аналіз) використовується для виявлення статистичних зв'язків, проведення гіпотезних тестів, розрахунку показників центральної тенденції, варіації та інших статистичних метрик. Інструменти для статистичного аналізу, такі як R, Python (з використанням бібліотек, таких як NumPy та Pandas) та SPSS, можуть бути використані для цих цілей.

Machine Learning (машинне навчання) використовується для створення моделей, які можуть прогнозувати, класифікувати або здійснювати інші дії на основі даних. Воно може бути застосоване для аналізу даних у базі даних та виявлення складних залежностей. Інструменти для машинного навчання, такі як TensorFlow, scikit-learn та Keras, можуть бути використані для цих цілей.

Ці методи та інструменти є лише деякими з багатьох доступних для аналізу даних у базах даних. Вибір конкретних методів та інструментів залежить від потреб, типу даних та масштабу проєкту.

1.4 Аналіз стратегій масштабування БД

Стратегії масштабування баз даних включають вертикальне та горизонтальне масштабування.

Вертикальне масштабування (Scale-up) включає збільшення потужності апаратного забезпечення або ресурсів на одній фізичній машині. Це може включати оновлення процесора, додавання більшої кількості оперативної пам'яті або використання більш швидкодіючих сховищ даних. При цьому вся база даних знаходиться на одній фізичній машині.

Переваги вертикального масштабування:

- простота у впровадженні, оскільки не потрібно налаштовувати складну інфраструктуру з кластерами;
- забезпечує високу продуктивність для невеликих та середніх навантажень;

– знижує потребу у зайнятості великої кількості серверів.

Обмеження вертикального масштабування:

– існує межа фізичних ресурсів, яку можна використовувати на одній машині;
– зростання вартості збільшення потужності апаратного забезпечення.

Горизонтальне масштабування (Scale-out) включає розподіл бази даних на кілька фізичних машин, які працюють разом як кластер. Кожен сервер в кластері має свою копію даних або певну частину даних. При такому масштабуванні навантаження розподіляються між серверами, що дозволяє обробляти більше запитів та забезпечує більшу масштабованість.

Переваги горизонтального масштабування:

– забезпечує високу масштабованість, оскільки можна додавати нові сервери до кластеру;
– забезпечує вищу доступність, оскільки при відмові одного сервера інші можуть продовжувати працювати;
– можливість географічної реплікації даних.

Обмеження горизонтального масштабування:

– складніше впровадження та управління кластером серверів;
– деякі операції, такі як з'єднання даних з різних серверів, можуть бути складнішими.

Вибір між вертикальним і горизонтальним масштабуванням залежить від потреб, обсягу даних, прогнозованого навантаження та бюджету. Деякі проєкти можуть використовувати комбінацію обох стратегій, де окремі компоненти бази даних масштабуються вертикально, а весь кластер масштабується горизонтально.

1.5 Аналіз використання БД у застосунках

Бази даних в застосунках є важливим компонентом для зберігання та управління даними, які використовуються в програмах та застосунках.

Управління користувачами та автентифікація – БД може використовуватись для зберігання інформації про користувачів, їх облікові записи, паролі та інші дані, пов'язані з автентифікацією та авторизацією. Це дозволяє застосунку перевіряти права доступу користувачів та забезпечувати безпеку.

БД може використовуватись для зберігання конфігураційних даних, таких як налаштування програми, параметри підключення до зовнішніх сервісів або інші налаштування, які можуть змінюватись в залежності від потреб застосунку.

База даних може використовуватись для зберігання основних даних, що використовуються у застосунку, таких як інформація про продукти, замовлення, клієнтів, повідомлення тощо. Застосунок може виконувати операції вставки, оновлення, видалення та вибірки даних з бази даних для обробки та відображення користувачу.

Аналітика та створення звітів – БД може використовуватись для зберігання даних. Це може включати статистичні дані, тренди, показники продуктивності та інші метрики, які допомагають в оцінці ефективності застосунку та прийнятті рішень.

База даних може використовуватись для синхронізації та реплікації даних між різними екземплярами застосунку або між різними серверами. Це дозволяє забезпечити доступ до актуальних даних та забезпечити високу доступність та надійність системи.

Інтеграція зовнішніх сервісів та джерел даних – БД може використовуватись для зберігання та обробки даних, отриманих зовнішніми сервісами або джерелами, такими як API, веб-служби, інші бази даних тощо. Це дозволяє застосунку отримувати, обробляти та зберігати дані з різних джерел.

2 АНАЛІЗ ЕТАПІВ СТВОРЕННЯ БД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ІСНУЮЧИХ СУБД

Для створення бази даних в тому числі БД для генерації тестових білетів слід визначити структуру даних і вибрати відповідну систему управління базами даних (СУБД).

Перший етап це визначення структуру даних. На цьому етапі, слід розглянути, яку інформацію потрібно зберігати для кожного білету. Наприклад, це може бути текст питання, варіанти відповідей, правильна відповідь і т. д. Розбити цю інформацію на відповідні таблиці та поля.

Наступний етап – вибір СУБД. Існує безліч СУБД, таких як MySQL, PostgreSQL, SQLite та інші. Вибір залежить від того, яка найбільше СУБД підходить для проєкту з урахуванням вимог до продуктивності, масштабованості та доступності.

Третій етап – створення таблиць. Використовуючи обрану СУБД, необхідно створити таблиці, що відповідають структурі даних. Кожна таблиця повинна мати поля, які відповідають різним атрибутам білета, а також первинний ключ для унікальної ідентифікації кожного білета.

Далі, етап заповнення таблиці даними. Введення даних в таблиці, використовуючи мову SQL або інструменти керування базами даних. Дані можна безпосередньо вводити в таблиці або завантажувати їх із файлу.

На п’ятому етапі, необхідно написати запити для генерації тестових білетів. Використовуючи мову SQL, можна написати запити, які витягнуть необхідну інформацію з таблиць для створення тестових білетів. Наприклад, можна написати запит, який вибере випадкові питання з бази даних та їх відповідні варіанти відповідей.

Шостий етап передбачає, розробку програми для генерації білетів (опціонально). Якщо планується створення застосунку для генерації тестових білетів, потрібно буде розробити відповідний програмний інтерфейс (API), який буде взаємодіяти з базою даних та генерувати квитки на основі запитів.

На сьомому етапі відбувається тестування та оптимізація БД. На цьому етапі, варто переконатися, що розроблена БД працює належним чином та відповідає на запити ефективно. Якщо потрібно, слід провести оптимізацію запитів або структури БД для підвищення продуктивності.

Також, слід звернути увагу, що створення бази даних та генерація тестових білетів – це складний процес, і при розробці може знадобитися додаткове вивчення, особливо в галузі мови SQL та обраної СУБД.

Вибір СУБД залежить від декількох факторів, включаючи вимоги, переваги та контекст проєкту. На сьогоднішній день одними з найпопулярніших СУБД можна вважати MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle та MongoDB.

MySQL (рис.2) є однією з найпопулярніших СУБД з відкритим вихідним кодом. Вона добре масштабується, має високу продуктивність і широке співтовариство підтримки.

Рисунок 2 – Логотип СУБД MySQL

Переваги використання MySQL:

- широка популярність та велика спільнота підтримки;
- відмінна продуктивність для обробки великого обсягу даних;
- простота встановлення та використання;
- хороша масштабованість, підтримка реплікації та кластеризації.

До недоліків цієї СУБД, слід віднести:

- обмежена підтримка складних структур даних, таких як JSON або XML;
- відсутність вбудованої підтримки розподіленої обробки даних та масштабування;
- недостатня підтримка транзакцій для деяких складних операцій;
- обмеження щодо продуктивності в обробці великих обсягів даних.

PostgreSQL (див.рис.3) також є потужною СУБД з відкритим вихідним кодом. Вона підтримує складні запити, має широкий набір функцій та відома своєю надійністю.

Рисунок 3 – Логотип СУБД PostgreSQL

Переваги використання PostgreSQL:

- відмінна підтримка складних запитів та розширених функцій, включаючи геодані, повнотекстовий пошук та інші;
- висока надійність та цілісність даних;

– розширюваність і можливість створення власних типів даних і функцій;
– хороша підтримка транзакцій та багатопоточності .

До недоліків цієї СУБД, слід віднести:

– вимоги до адміністрування та конфігурації можуть бути високими;
– періодичні випуски нових версій, які можуть вимагати оновлення та тестування існуючих систем;
– відсутність підтримки деяких популярних комерційних інструментів, які більше орієнтовані на інші СУБД.

SQLite є полегшеною СУБД, що вбудовується, вона не вимагає окремого сервера. Дана СУБД (рис.4) підходить для невеликих проєктів та програм, що працюють на одному пристрої.

Рисунок 4 – Логотип СУБД SQLite

Переваги використання SQLite:

– легкість і простота вбудовування не потребує окремого сервера;
– нульове адміністративне навантаження та простота використання;
– підтримка більшості мов програмування;
– ідеально підходить для невеликих проектів та вбудованих систем.

До недоліків цієї СУБД, слід віднести:

- обмежена масштабованість (SQLite не призначена для роботи з великими обсягами даних або високонавантаженими програмами; вона найефективніша при обробці невеликих баз даних, які можуть вміститися в оперативній пам'яті);
- однокористувацька (SQLite призначена для однокористувацького використання і не підтримує одночасний доступ до бази даних кількома користувачами; це обмеження може бути непридатним для сценаріїв, де потрібний паралельний доступ до даних для багатьох користувачів);
- відсутність віддаленого доступу (SQLite не надає вбудованого механізму віддаленого доступу до бази даних; це означає, що під час використання SQLite дані повинні бути доступні локально на тому ж комп'ютері або пристрої, де запущено програму);
- обмежена функціональність (SQLite пропонує базовий набір функціональності порівняно з іншими СУБД; наприклад, вона не підтримує деякі розширені можливості, такі як процедури, що зберігаються, тригери або реплікацію даних);
- немає централізованого сервера (SQLite не використовує централізований сервер для керування базою даних; це може означати, що необхідно самостійно керувати та оновлювати базу даних, включаючи резервне копіювання та відновлення даних).

Якщо розробка пов’язана з екосистемою Microsoft, SQL Server (рис.5) може бути хорошим вибором. Вона пропонує широкий набір функцій, інструментів та підтримку з боку Microsoft.

Рисунок 5 – Логотип СУБД Microsoft SQL Server

Переваги використання Microsoft SQL Server:

- великий набір функцій та інструментів розробки, особливо в екосистемі Microsoft;
- висока продуктивність і масштабованість, особливо на платформі Windows ;
- підтримка шифрування та забезпечення безпеки даних;
- хороша інтеграція з іншими продуктами Microsoft, такими як .NET та Azure.

До недоліків цієї СУБД, слід віднести:

- обмежена кросплатформеність, оскільки Microsoft SQL Server підтримується переважно на Windows-сервера;
- вартість ліцензій та підтримки може бути високою для підприємств з обмеженими бюджетами;
- деякі функції та можливості доступні тільки в платній версії SQL Server Enterprise Edition.

Oracle є однією з найпотужніших комерційних СУБД із широкими можливостями. Вона часто використовується у великих підприємствах та проектах, де потрібна висока продуктивність та масштабованість (див.рис.6).

Рисунок 6 – Логотип СУБД Oracle

Переваги використання Oracle:

– висока продуктивність і масштабованість, особливо під час роботи з великими обсягами даних;
– широкий набір функцій та можливостей, включаючи управління даними, аналітику та бізнес-інтелект;
– чудова надійність та відмовостійкість;
– широка підтримка корпоративних клієнтів та великий досвід роботи з великими проектами.

До недоліків цієї СУБД, слід віднести:

– висока вартість ліцензій та підтримки Oracle Database;
– велика кількість функцій і можливостей, що може призводити до складності у використанні та адмініструванні;
– важкість масштабування та розподіленої обробки даних;
– великі вимоги до обладнання та ресурсів для ефективної роботи.

MongoDB (от англ. "humongous" – величезний) – це документ-орієнтована Система Управління Базами Даних (СУБД), яка використовує модель даних, відому як BSON (Binary JSON). Вона розроблена для зберігання та обробки великих обсягів структурованих, напівструктурованих і навіть неконструктивних даних (рис.7).

Рисунок 7 – Логотип СУБД MongoDB

Переваги використання MongoDB:

– гнучка схема даних (MongoDB дозволяє зберігати дані без фіксованих структур, що дозволяє зручно працювати з нерегулярною або змінюваною структурою даних; це особливо корисно, коли потрібно працювати з невизначеними або еволюційними даними);
– масштабованість та висока продуктивність (MongoDB підтримує горизонтальне масштабування, що дозволяє розподіляти дані на кілька серверів і обробляти великі обсяги даних та високе навантаження; вона також добре працює з розподіленими середовищами, такими як хмарні обчислення);
– потужна мова запитів (MongoDB надає потужну запитову мову, яка дозволяє виконувати складні операції з даними; запитова мова MongoDB підтримує широкий спектр операцій, таких як фільтрація, сортування, агрегація, групування та багато інших);
– географічна реплікація та шардування (MongoDB дозволяє розподіляти дані на кілька фізичних вузлів за допомогою механізму шардування; це дозволяє забезпечити високу доступність та стійкість до відмов);
– вбудована підтримка для роботи з геопросторовими даними (MongoDB має вбудовану підтримку для роботи з геопросторовими даними та може виконувати запити, пов'язані з геолокацією, такі як пошук найближчих об'єктів або операції з полігонами);
– простота використання та розробки (MongoDB має зрозумілий синтаксис та API, що спрощує розробку та інтеграцію з додатками; вона також надає багато інструментів та документацію для полегшення роботи з базою даних).

До недоліків цієї СУБД, слід віднести:

– відсутність вбудованої підтримки транзакцій у попередніх версіях (хоча останні версії MongoDB включають підтримку транзакцій);

- вища вимога до ресурсів, особливо при роботі з великими обсягами даних;
- обмежена підтримка складних SQL-запитів та зв'язків між колекціями даних.

Вибір конкретної СУБД залежить від переваг, доступних ресурсів, вимог до масштабованості, продуктивності, безпеки та підтримки. Рекомендується провести додаткове дослідження та порівняння різних СУБД для вибору найбільш підходящої для проєкту.

Таблиця 1 – Порівняння характеристик найпопулярніших СУБД

СУБД	MySQL	PostgreSQL	SQLite	Microsoft SQL Server	Oracle
Тип ліцензії	Open Source	Open Source	Public Domain	Комерційна	Комерційна
Операційні системи	Кросплатформова	Кросплатформова	Кросплатформова	Windows	Кросплатформова
Мова програмування	Безліч	Безліч	Безліч	Безліч	Безліч
Підтримка SQL	+	+	+	+	+
Масштабованість	Гарна	Відмінна	Обмежена	Відмінна	Відмінна
Надійність	Гарна	Відмінна	Гарна	Відмінна	Відмінна
Продуктивність	Висока	Висока	Помірна	Висока	Висока
Спільнота підтримки	Широке	Широке	Помірне	Широке	Широке

3 ОПИС РОЗРОБКИ БД ДЛЯ ФОРМУВАННЯ БІЛЕТІВ З ФАХОВОГО ІСПИТУ

База даних призначена для формування білетів з фахового іспиту а також для зберігання та використання даних про студентів які проходитимуть фаховий іспит. Предметна область складається із чотирьох сутностей «student», «test», «questions» і «test_has_student», вони використовуються для зберігання та організації даних, пов'язаних зі студентами, тестами та питаннями.

Для створення таблиці «questions» було написано наступний SQL-запит:

```
CREATE TABLE IF NOT EXISTS `juniorbachelor`.`questions` (
  `id_questions` INT NOT NULL,
  `questions_text` VARCHAR(255) NULL,
  `answer_1` VARCHAR(255) NULL,
  `answer_2` VARCHAR(255) NULL,
  `answer_3` VARCHAR(255) NULL,
  `answer_4` VARCHAR(255) NULL,
  `correct_answ` VARCHAR(255) NULL,
  `subject_name` VARCHAR(45) NULL,
  `select_answ` VARCHAR(255) NULL,
  `url_img` VARCHAR(45) NULL,
  `id_test` INT NULL,
  PRIMARY KEY (`id_questions`),
  CONSTRAINT `id_test`
    FOREIGN KEY (`id_test`)
    REFERENCES `juniorbachelor`.`test` (`id_test`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,)
ENGINE = InnoDB
```

Для створення таблиці «questions» було написано наступний SQL-запит:

```
CREATE TABLE IF NOT EXISTS `juniorbachelor`.`student` (
  `id_student` INT NOT NULL,
  `result` INT NULL,
  `id_test` INT NULL,
  `name` VARCHAR(45) NULL,
  `surname` VARCHAR(45) NULL,
  PRIMARY KEY (`id_student`),
  INDEX `result_idx` (`result` ASC) VISIBLE,
  INDEX `id_test_idx` (`id_test` ASC) VISIBLE,
  CONSTRAINT `id_test`
    FOREIGN KEY (`id_test`)
    REFERENCES `juniorbachelor`.`test` (`id_test`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

Для створення таблиці «test» було написано наступний SQL-запит:

```
CREATE TABLE IF NOT EXISTS `juniorbachelor`.`test` (
  `id_test` INT NOT NULL,
  `result` INT NOT NULL,
  `id_student` INT NULL,
  `duration` INT NULL,
  `data` INT NULL,
  `answ_q1` INT NULL,
     . . . . .
  `answ_q100` INT NULL,
  PRIMARY KEY (`id_test`, `result`),
  INDEX `id_student_idx` (`id_student` ASC) VISIBLE,
  CONSTRAINT `id_student`
    FOREIGN KEY (`id_student`)
    REFERENCES `juniorbachelor`.`student` (`id_student`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

Для створення таблиці «test» було написано наступний SQL-запит:

```
CREATE TABLE IF NOT EXISTS `juniorbachelor`.`test_has_student` (
  `test_id_test` INT NOT NULL,
  `student_id_student` INT NOT NULL,
  `student_result` INT NOT NULL,
  PRIMARY KEY (`test_id_test`, `student_id_student`),
  INDEX `fk_test_has_student_student1_idx` (`student_id_student`
ASC) VISIBLE,
  INDEX `fk_test_has_student_test1_idx` (`test_id_test` ASC)
VISIBLE,
  INDEX `fk_test_has_student_result_idx` (`student_result` ASC)
VISIBLE,
  CONSTRAINT `fk_test_has_student_test`
    FOREIGN KEY (`test_id_test`)
    REFERENCES `juniorbachelor`.`test` (`id_test`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_test_has_student_student`
    FOREIGN KEY (`student_id_student`)
    REFERENCES `juniorbachelor`.`student` (`id_student`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_test_has_student_result`
    FOREIGN KEY (`student_result`)
    REFERENCES `juniorbachelor`.`test` (`result`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

Нижче представлений короткий опис кожної таблиці та їх полів.

Таблиця «student» складається з таких полів як:

- id_student: унікальний ідентифікатор студента;
- result_st: результат проходження тесту студентом;
- id_test: унікальний ідентифікатор тесту;

– name: ім’я студента;
– surname: прізвище студента.

Таблиця «test»:

– id_test: унікальний ідентифікатор тесту.
– result: результат проходження тесту студентом;
– id_student: унікальний ідентифікатор студента;
– duration: тривалість тест;
– data: дата складання/проходження тесту;
– answ_q1 – answ_q100: відповіді студента на питання 1 – 100.

Таблиця «questions»:

– id_questions: унікальний ідентифікатор питання;
– questions_text: текст питання;
– answer_1, answer_2, answer_3, answer_4: варіанти відповідей на питання;
– correct_answ: позначення правильної відповіді на питання;
– subject_name: назва предмету до якого належить питання;
– select_answ: коментар до питання;
– url_img: посилання на рисунок;
– id_test: зовнішній ключ, який посилається на ідентифікатор тесту, до якого належить питання.

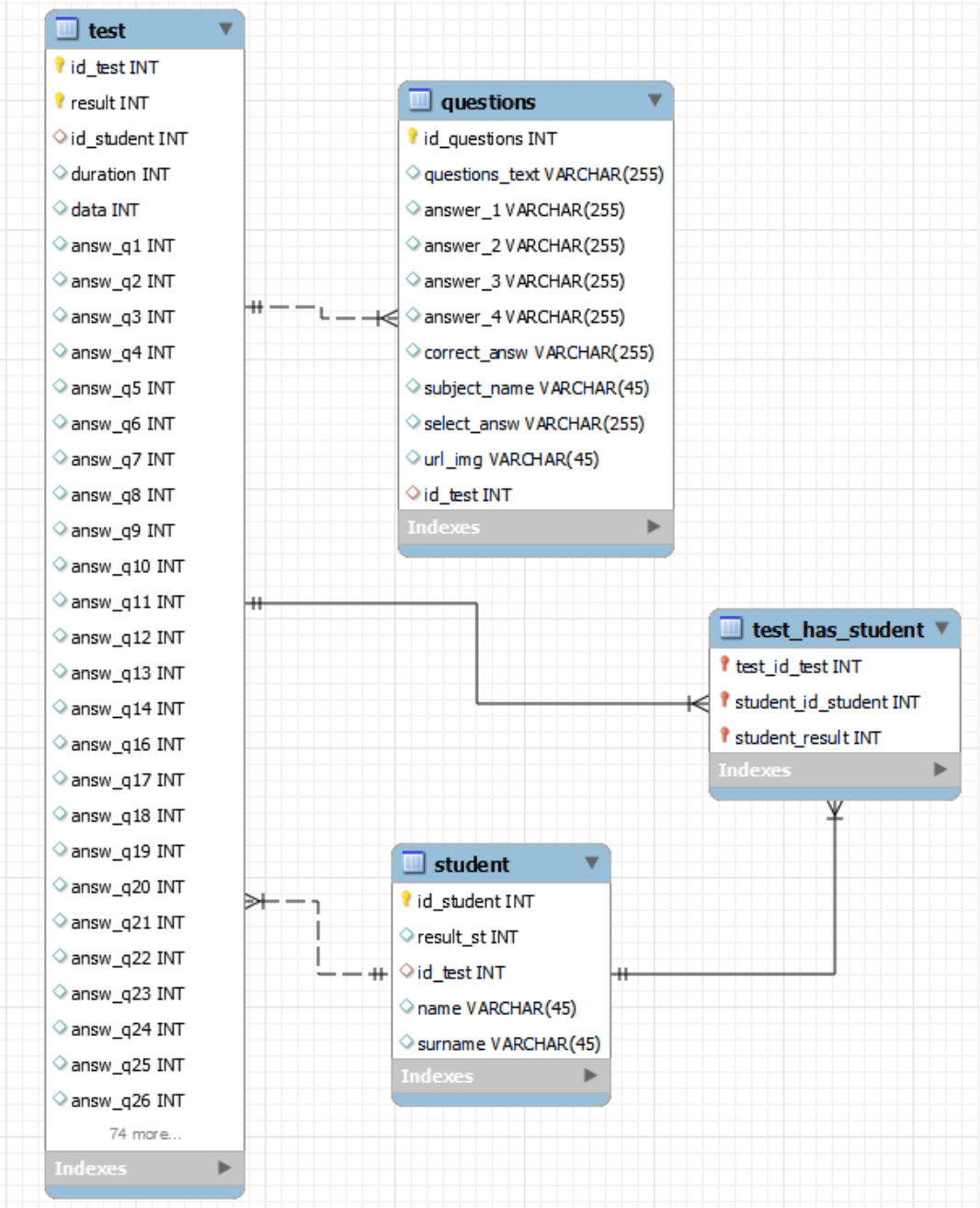

Рисунок 8 – Реляційна модель даних БД «juniorbachelor»

Ці таблиці (рис. 8) можуть мати зв'язки між собою. Наприклад, поле «id_test» в таблиці «questions» може посилатися на поле «id_test» в таблиці «test», щоб встановити зв'язок між питаннями і тестами. Аналогічно, можна мати зв'язок між таблицями «student» і «test», якщо, наприклад, студенти по-в'язані зі своїми пройденими тестами.

Залежно від конкретної потреби, можна додати додаткові поля в таблиці або створити інші таблиці, які відображають більш специфічні вимоги щодо даних. Важливо належним чином спроєктувати базу даних, визначивши правильні зв'язки і поля, щоб забезпечити ефективне зберігання і маніпуляцію даними. База даних була створена на базі СУБД MySQL [4] за допомогою застосунку MySQL Workbench (рис.9).

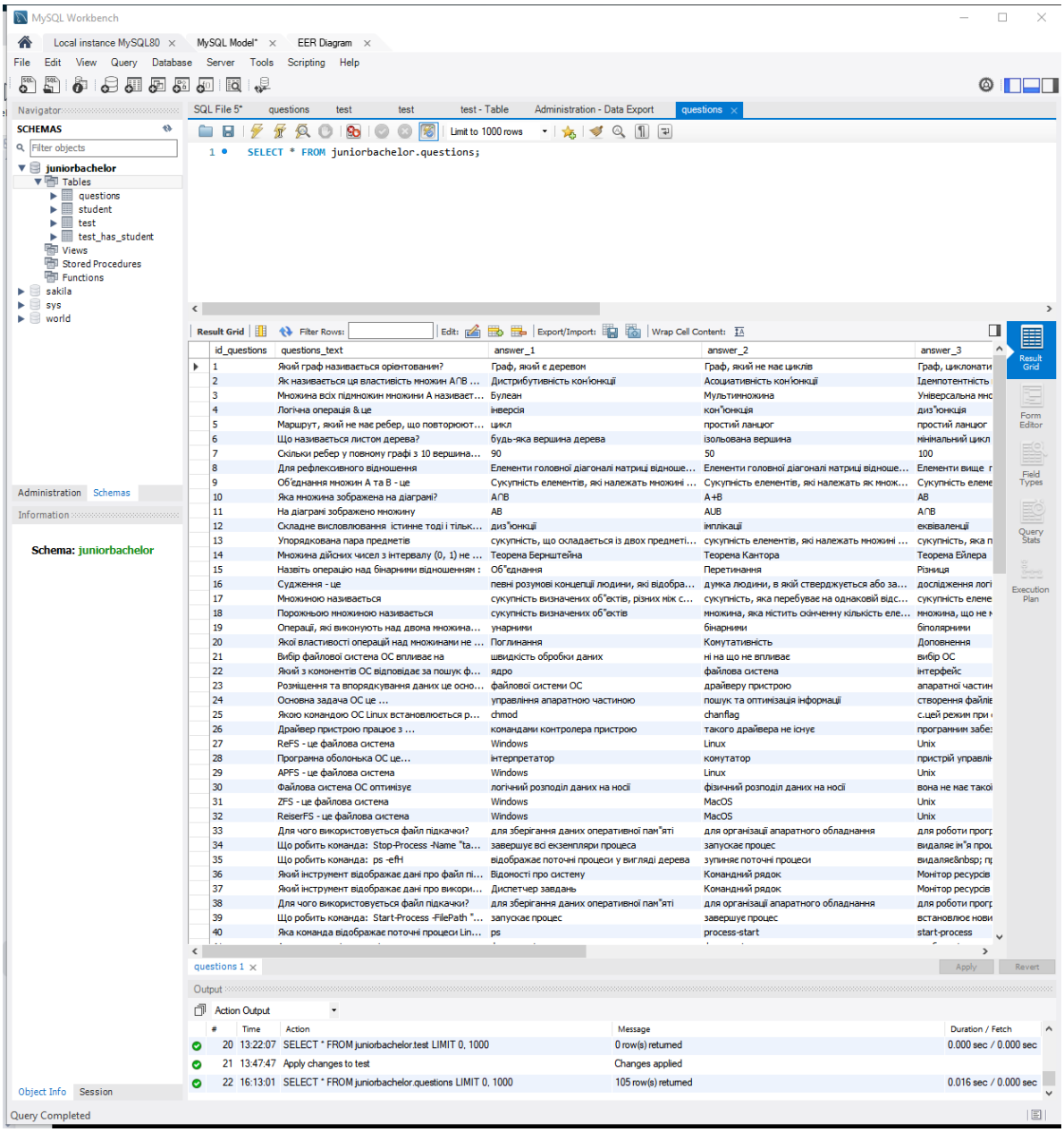

Рисунок 9 – Середовище застосунку MySQL Workbench

Запити, які можна виконати до цих таблиць, залежать від конкретної потреби та функціональності системи. Нижче описані деякі типові запити, які можна зробити до таблиць «student», «test» та «questions».

Запит «вибрати всіх студентів»:

```
SELECT * FROM juniorbachelor.student;
```

Запит «вибрати конкретного студента за його ідентифікатором»:

```
SELECT * FROM juniorbachelor.student WHERE id_student = <id>;
```

Запит «вибрати всі тести»:

```
SELECT * FROM juniorbachelor.test;
```

Запит «вибрати питання для певного тесту за його ідентифікатором»:

```
SELECT * FROM juniorbachelor.questions WHERE id_test = <id>;
```

Запит «додати нового студента до бази даних»:

```
INSERT INTO juniorbachelor.student (id_student, name, surname, result_st, [інші поля]) VALUES (<id>, '<ім'я>', '<прізвище>', '<результат_тесту>', [інші значення]);
```

Запит «оновити інформацію про студента»:

```
UPDATE juniorbachelor.student SET name = '<нове_ім'я>', surname = '<нове_прізвище>' WHERE id_student = <id>;
```

Запит «видалити студента за його ідентифікатором»:

```
DELETE FROM juniorbachelor.student WHERE id_student = <id>;
```

Запит «знайти всі питання з певного предмету»:

```
SELECT * FROM juniorbachelor.questions WHERE subject_name = '<предмет>';
```

Запит «знайти кількість питань у певному тесті»:

```
SELECT COUNT(*) FROM juniorbachelor.questions WHERE id_test = <id>;
```

Запит «внесення нового питання до таблиці»:

```
INSERT INTO juniorbachelor.test (id_questions, questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ, subject_name, select_answ, url_img, id_test) VALUES (, '', '', '', '', '', '', '', '', '', '');
```

На рис. 10 представлено робоче середовище, в якому відбувається розробка. З представленого скріншоту, можна побачити початкові команди, які застосовувались для наповнення таблиці «questions», тобто заповнення бази даними відповідними питаннями з вказанням дисциплін до яких відносяться дані питання. Як висновок, можна зауважити, що до БД «juniorbachelor», а точніше до її таблиці «questions» було додано більше ніж 100 питань з 5 профільних дисциплін, які викладаються на спеціальності Комп’ютерні науки. Після даного рисунку, представлено уривок коду, який використовувався для запису даних до таблиці.

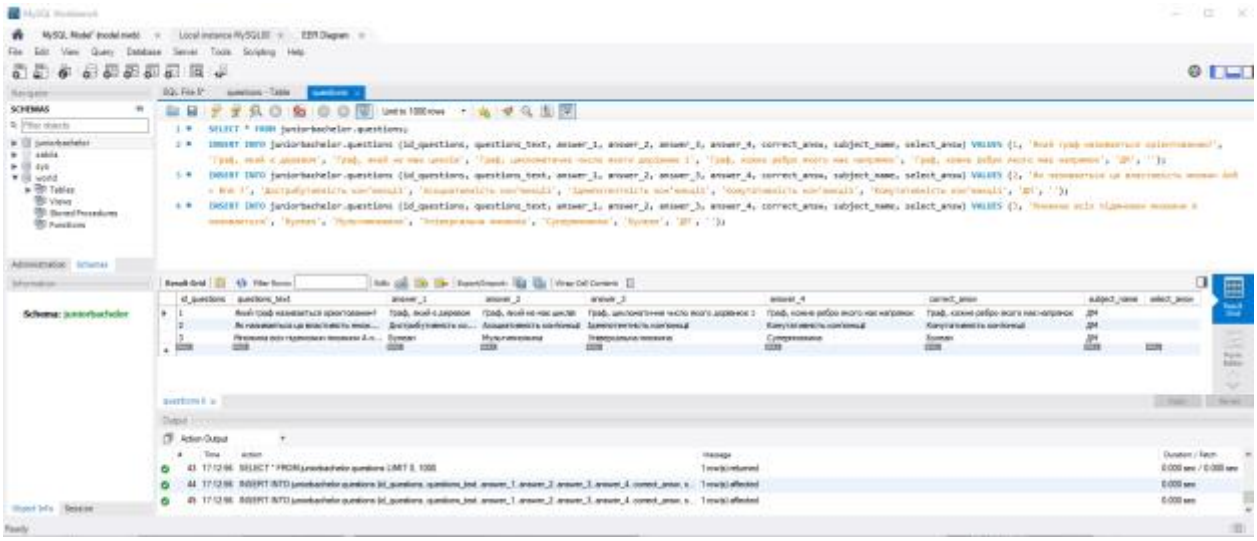

Рисунок 10 – Середовище застосунку MySQL Workbench

```
INSERT INTO juniorbachelor.questions (id_questions, questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ, subject_name, select_answ) VALUES (1, 'Який граф називається орієнтованим?', 'Граф, який є деревом', 'Граф, який не має циклів', 'Граф, цикломатичне число якого дорівнює 1', 'Граф, кожне ребро якого має напрямок', '', 'ДМ', '');
INSERT INTO juniorbachelor.questions (id_questions, questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ, subject_name, select_answ) VALUES (2, 'Як називається ця властивість множин A∩B = B∩A ?', 'Дистрибутивність кон’юнкції', 'Асоциативність
```

```
кон’юнкції', 'Ідемпотентність кон’юнкції', 'Комутативність кон’юнкції',
'', 'ДМ', '');
     INSERT     INTO     juniorbachelor.questions     (id_questions,
questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ,
subject_name, select_answ) VALUES (3, 'Множина всіх підмножин множини
A називається', 'Булеан', 'Мультимножина', 'Універсальна множина',
'Супермножина', '', 'ДМ', '');
     INSERT     INTO     juniorbachelor.questions     (id_questions,
questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ,
subject_name, select_answ) VALUES (4, 'Логічна операція & це', 'інвер-
сія', 'кон"юнкція', 'диз"юнкція', 'немає вірної', '', 'ДМ', '');
     INSERT     INTO     juniorbachelor.questions     (id_questions,
questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ,
subject_name, select_answ) VALUES (5, 'Маршрут, який не має ребер, що
повторюються, це', 'цикл', 'простий ланцюг', 'простий ланцюг', 'лан-
цюг', '', 'ДМ', '');
     INSERT     INTO     juniorbachelor.questions     (id_questions,
questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ,
subject_name, select_answ) VALUES (6, 'Що називається листом дерева?',
'будь-яка вершина дерева', 'ізольована вершина', 'мінімальний цикл',
'висяча вершина', '', 'ДМ', '');
     INSERT     INTO     juniorbachelor.questions     (id_questions,
questions_text, answer_1, answer_2, answer_3, answer_4, correct_answ,
subject_name, select_answ) VALUES (7, 'Скільки ребер у повному графі з
10 вершинами?', '90', '50', '100', '45', , 'ДМ', '');
```

Для отримання кількості студентів, які набрали за тест більше 60 балів, можна скористатися запитом SQL з умовою WHERE. Припустимо, що є таблиця «student» зі стовпцями «id_student» і «result_st» . Ось приклад запиту:

```
SELECT COUNT(*) AS total_students
FROM juniorbachelor.student
WHERE result_st > 60;
```

Цей запит підрахує кількість записів, в яких значення стовпця «result_st» перевищує 60, і поверне результат у стовпці «total_students».

Для отримання кількості студентів, які пройшли тест в певний день, потрібно мати таблицю, в якій зберігаються дані про тести та результати студентів. Припустимо, що є таблиця «test» зі стовпцями «id_student», «result» та «data». Ось приклад запиту:

```
SELECT COUNT(*) AS total_students
FROM juniorbachelor.test
WHERE data = 'певний день' AND result_st >= 60;
```

У цьому запиті «певний день» потрібно замінити на конкретну дату, для якої слід отримати кількість студентів. «result» може бути відповідним значенням для позначення пройденого тесту. Запит підрахує кількість записів, в яких значення стовпця «data» співпадає з вказаною датою і значення стовпця «result» відповідає позначенню пройденого тесту, і поверне результат у стовпці «total_students».

Враховуючи інформацію, яка наведено на початку розділу, а саме основні таблиці включають «student» (студент), «test» (тест) та «questions» (питання), основні поля для кожної таблиці.

На основі цієї інформації можна зробити деякі висновки, а саме для встановлення зв’язків між таблицями, були використали зовнішні ключі. Наприклад, поле «id_student» у таблиці «student» використовується як зовнішній ключ у таблиці «test» для зв'язку з результатами студента. Так само, поле «id_test» у таблиці «questions» використовується як зовнішній ключ для зв'язку з конкретним тестом, до якого належить питання. Це допомагає побудувати зв'язану структуру бази даних і забезпечити цілісність даних. Що стосується множинності відповідей, можна звернути увагу на таблицю «questions» є поля для варіантів відповідей на питання («answer_1», «answer_2», «answer_3», «answer_4»). Це вказує на можливість, що одне питання може мати кілька варіантів відповідей, і користувач може вибрати один з них. У вигляді додаткової інформація, слід приділити увагу таблиці

«questions" також є поля, які містять додаткову інформацію про питання, таку як коментарі («select_answ»), посилання на рисунок («url_img») та назва предмету («subject_name»). Ці поля можуть бути корисними для подальшого аналізу даних або відображення додаткової інформації в інтерфейсі користувача.

Потрібно зазначити, що розроблена БД для формування білетів для тесту має свої переваги, а саме:

- ефективне зберігання даних (БД дозволяє ефективно зберігати і керувати великою кількістю питань, варіантів відповідей та інших відомостей, що стосуються тесту);
- швидкий доступ до даних (БД може оптимізувати запити і запити до даних, що дозволяє швидко отримувати результати тестування і формувати білети);
- гнучкість і розширюваність (БД може бути легко розширена і модифікована для додавання нових питань, варіантів відповідей або інших компонентів тесту, що дозволяє гнучко адаптувати систему до змінних вимог і потреб користувачів);
- централізоване керування даними (БД дозволяє централізовано керувати всіма даними, пов'язаними з тестом, це спрощує процес збереження, оновлення і управління даними та забезпечує їх консистентність);
- забезпечення цілісності даних (БД може використовувати механізми перевірки цілісності, щоб гарантувати правильність та консистентність даних, наприклад, перевіряти правильність відповідей на питання).

Обмеження розробленої бази даних для формування білетів на тест можуть включати:

- вимога до ресурсів (БД може вимагати значних ресурсів, таких як обсяг дискового простору і потужність обчислювальних систем, особливо при роботі з великою кількістю даних);

– залежність від надійності бази даних (якщо база даних виникає проблем).

Також можна виділити деякі особливості розробленої бази даних для формування білетів на тест, до них можна включати наступне.

Таблиця «student» зберігає інформацію про студентів, таку як їх ім'я, прізвище та результати проходження тесту. Ця таблиця містить зв'язок з таблицями «test» та «questions» за допомогою зовнішнього ключа.

Таблиця «test» містить дані про сам тест, включаючи його унікальний ідентифікатор, результати, тривалість, дату і зв'язок зі студентами, які проходили тест.

Таблиця «questions» містить інформацію про питання, включаючи текст питання, варіанти відповідей, правильні відповіді, коментарі, зображення та зв'язок з тестом, до якого вони належать.

Існує зв’язок між таблицями «test» та «student» за допомогою зовнішнього ключа «id_student», що дозволяє встановити зв'язок між студентами і їх результатами тесту.

Зв'язок між таблицями «questions» та «test» здійснюється за допомогою зовнішнього ключа «id_test», що дозволяє пов'язати питання із відповідним тестом.

Кожна таблиця має унікальний ідентифікатор, наприклад, «id_student»в таблиці «student», «id_test» в таблиці «test» та «id_questions» в таблиці «questions». Це дозволяє однозначно ідентифікувати записи в кожній таблиці.

База даних може включати перевірки цілісності, які гарантують правильність інформації, наприклад, перевірку правильності відповідей на питання та зв'язків між таблицями.

В таблиці «questions» зберігається детальна інформація про кожне питання, включаючи текст, відповіді, правильну відповідь та коментарі до питання.

База даних може бути легко розширена для додавання нових питань, тестів та студентів. За допомогою зв'язків між таблицями, нові дані можуть бути легко інтегровані в існуючу структуру бази даних.

Ці особливості бази даних для формування білетів на тест дозволяють зберігати і керувати відомостями про студентів, їх результатами тестування, питаннями та варіантами відповідей, що дозволяє ефективно проводити тестування та формувати білети для студентів.

Також, під час виконання даної кваліфікацйіної роботи бакалавра, було почато розробку для десктопного застосунку. Після аналізу інформації в мережі Інтернет було визначено, що для створення графічних інтерфейсів JavaFX є кілька способів, включаючи використання Scene Builder та написання коду вручну. Нижче описано кілька кроків, які можна виконати для створення графічного інтерфейсу JavaFX.

По-перше, це встановіть Java та JavaFX. Слід встановити Java Development Kit (JDK) і JavaFX. Завантажити JDK та JavaFX можна з офіційного веб-сайту Oracle.

По-друге, створення нового проєкту. У обраному інтегрованому середовищі розробки (наприклад, VS Code, IntelliJ IDEA або Eclipse) слід створити новий проєкт Java або JavaFX.

Далі, варто додати бібліотеку JavaFX. Для цього варто встановити налаштування проєкту таким чином, щоб вона включала бібліотеку JavaFX.

Четвертий етап, передбачає створіть головне вікно (Stage). У класі, що містить метод main, необхідно створити об'єкт Stage, який буде представляти головне вікно програми.

П’ятий етап, це створення кореневого вузла (Root Node). Необхідно визначити кореневий вузол (наприклад, Pane, GridPane, StackPane) для графічного інтерфейсу.

Далі, додати до кореневого вузла елементи інтерфейсу, сюди можна віднести різні елементи наприклад кнопки, мітки, тексти, поле введення тощо.

Шостий етап передбачає налаштування вигляду і розташування елементів. Варто використовувати методи і властивості JavaFX, щоб налаштувати вигляд і розташування елементів інтерфейсу.

Після, додаються обробники подій. Необхідно, налаштувати обробники подій для елементів інтерфейсу, щоб відповідати на дії користувача (натискання кнопок, введення даних тощо).

За допомогою використання методу show об'єкта Stage, можна показати головне вікно програми. Це загальна процедура створення графічного інтерфейсу JavaFX. Можна використовувати Scene Builder, щоб візуально створювати і редагувати інтерфейс, або написати код вручну, використовуючи класи та методи JavaFX. Scene Builder дозволяє швидко розташовувати елементи інтерфейсу та налаштовувати їх властивості без необхідності вручну писати код для цього.

Незалежно від способу, який обрано, важливо мати розуміння основних класів та методів JavaFX, що використовуються для створення графічних інтерфейсів, таких як Stage, Scene, Pane, Button, Label, TextField та багатьох інших.

Для розробки застосунку було обрано середовище VS Code та Scene Builder. VS Code і Scene Builder – це два інструменти, які часто використовуються в розробці програмного забезпечення. VS Code (Visual Studio Code) – це популярний текстовий редактор, розроблений компанією Microsoft. Він надає розширену функціональність для програмістів, зокрема підсвічування синтаксису, автодоповнення, вбудовану систему контролю версій, налаштування та розширення. VS Code підтримує багато мов програмування і надає зручне робоче середовище для написання коду. Scene Builder – це графічний інструмент, який використовується для візуального створення і редагування інтерфейсів користувача для JavaFX додатків. Він надає зручний інтерфейс для розташування елементів у вікні, налаштування їх властивостей та стилів. Scene Builder дозволяє розробникам швидше створювати складні інтерфейси без необхідності вручну писати код.

Отже, VS Code і Scene Builder використовуються для різних аспектів розробки програмного забезпечення. VS Code є загальним редактором коду, який можна використовувати для розробки в різних мовах програмування, в той час як Scene Builder є спеціалізованим інструментом для створення графічних інтерфейсів JavaFX додатків. Вони можуть використовуватись окремо або разом, залежно від потреб розробки конкретного проєкту. На рис.11 представлено середовище розробки, а також опис метода start що використовується для запуску першого шаблону який було розроблено за допомогою Scene Builder.

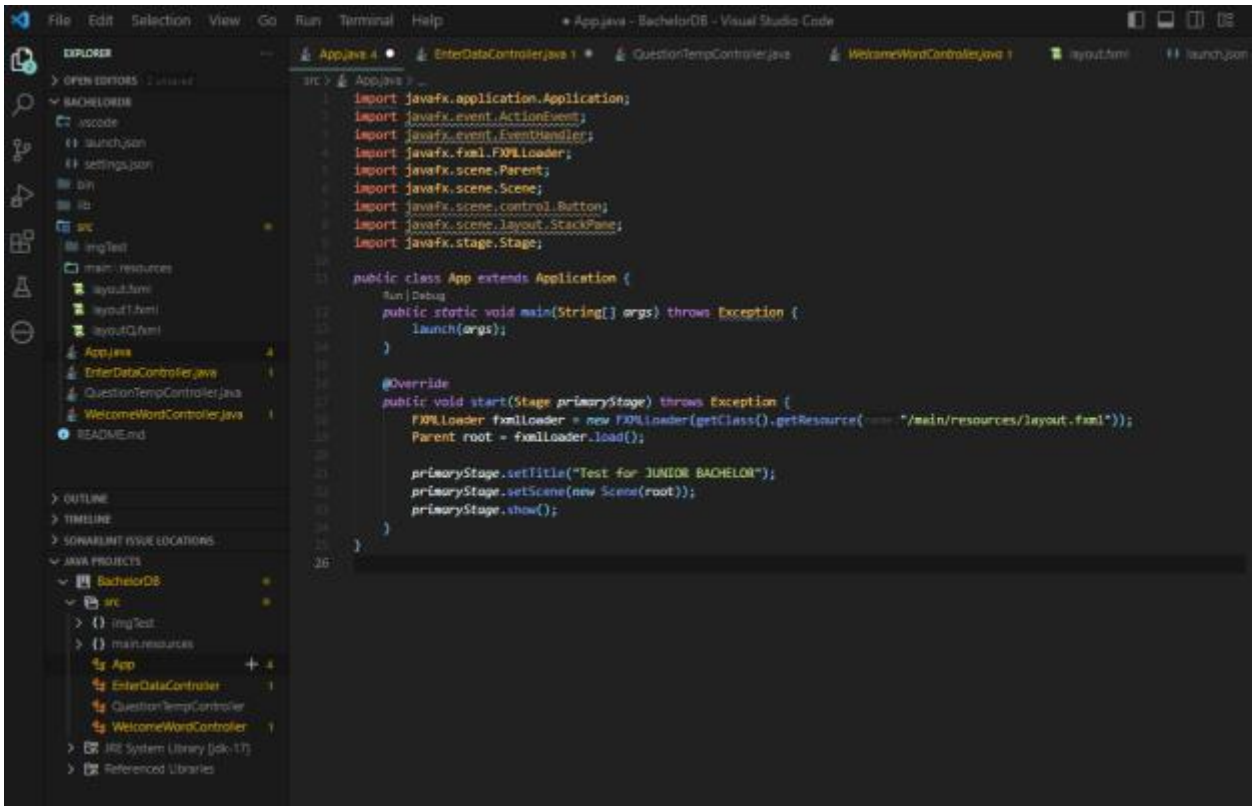

Рисунок 11 – Середовище застосунку VS Code

На рис.12 представлено головний екран, який буде показано студенту щойно застосунок буде запущено. Як можна побачити, даний екран складається з таких елементів як поля для вводу Ім’я та Прізвища студента, який буде проходити тестування. А також кнопка, щоб ввести дані.

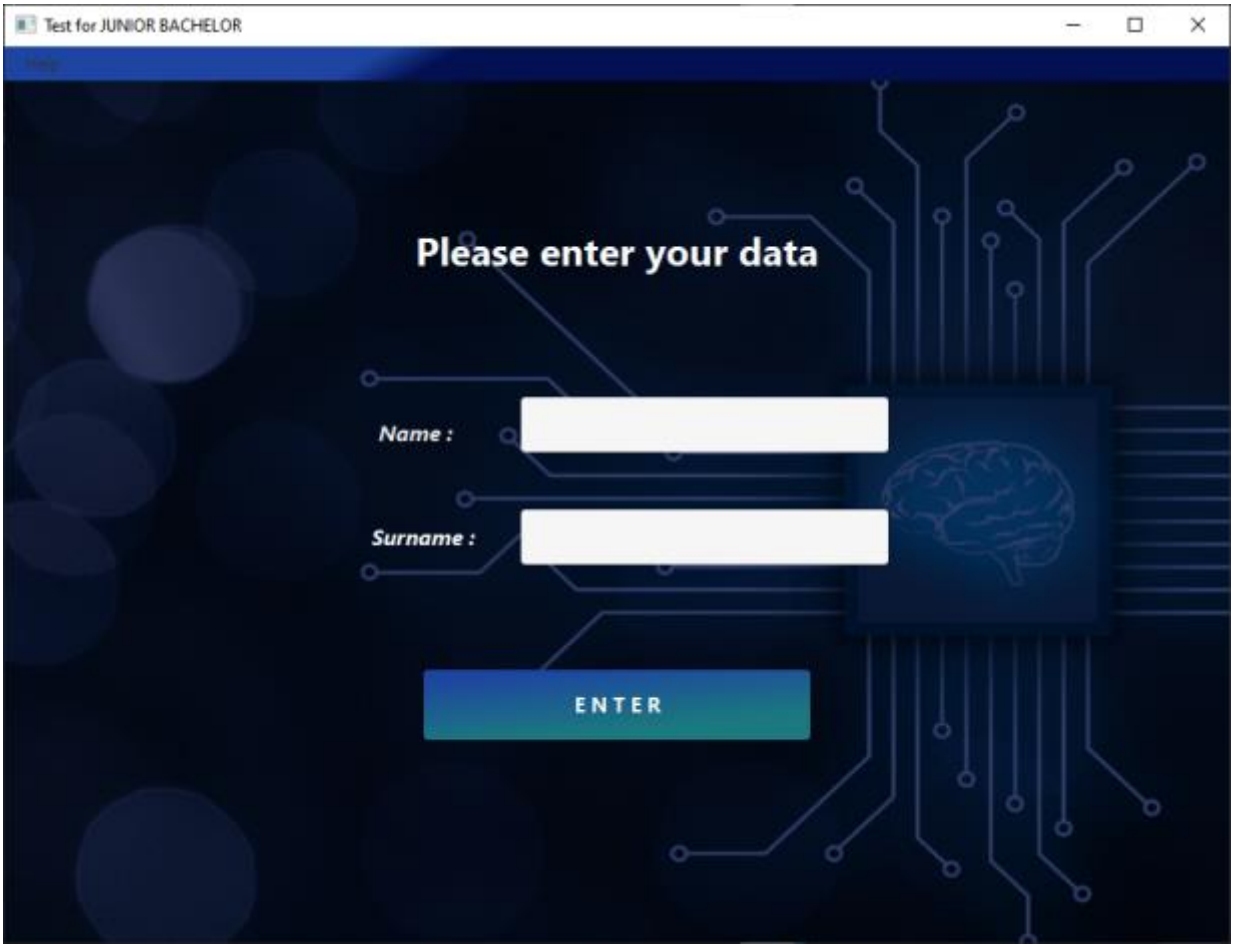

Рисунок 12 – Перший шаблон, для головного екрану застосунку

На рис.13 в свою чергу представлено другий екран, який буде показано студенту щойно студент натисне на кнопку Enter з попереднього екрану. А також кнопка, щоб розпочати тестування.

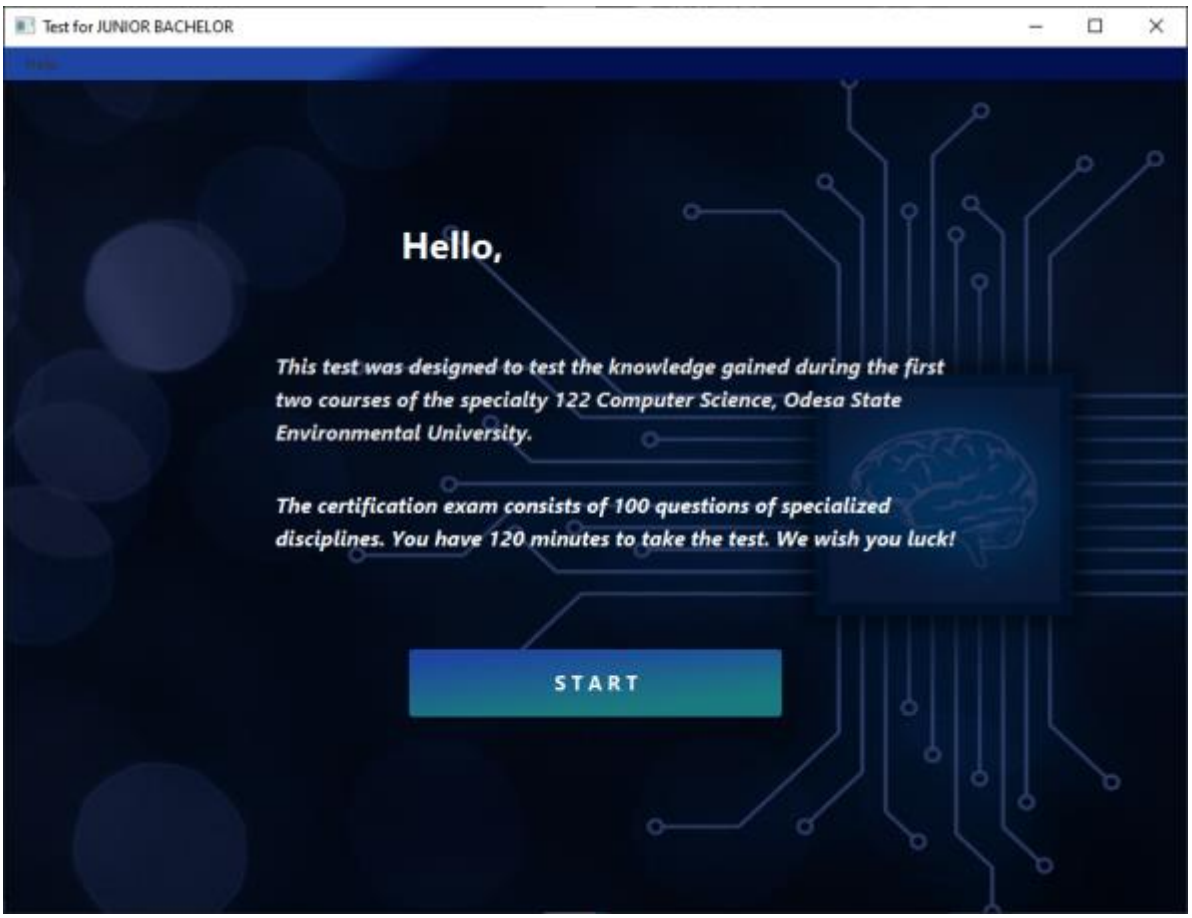

Рисунок 13 – Шаблон з вітальним словом яке представлено студенту який розпочне тестування

На рис.14 представлено третій екран, а саме це шаблон який в подальшому буде заповнюватись даними з БД. На даному шаблоні будуть виведені такі дані як:

- час що залишився до закінчення тестування;
- ім’я та прізвище студента, що проходить тестування;
- номер питання;
- текст питання;
- радіо-кнопки, які використовуватимуться для підставки/вибору правильної відповіді;
- кнопка переходу до наступного питання тесту.

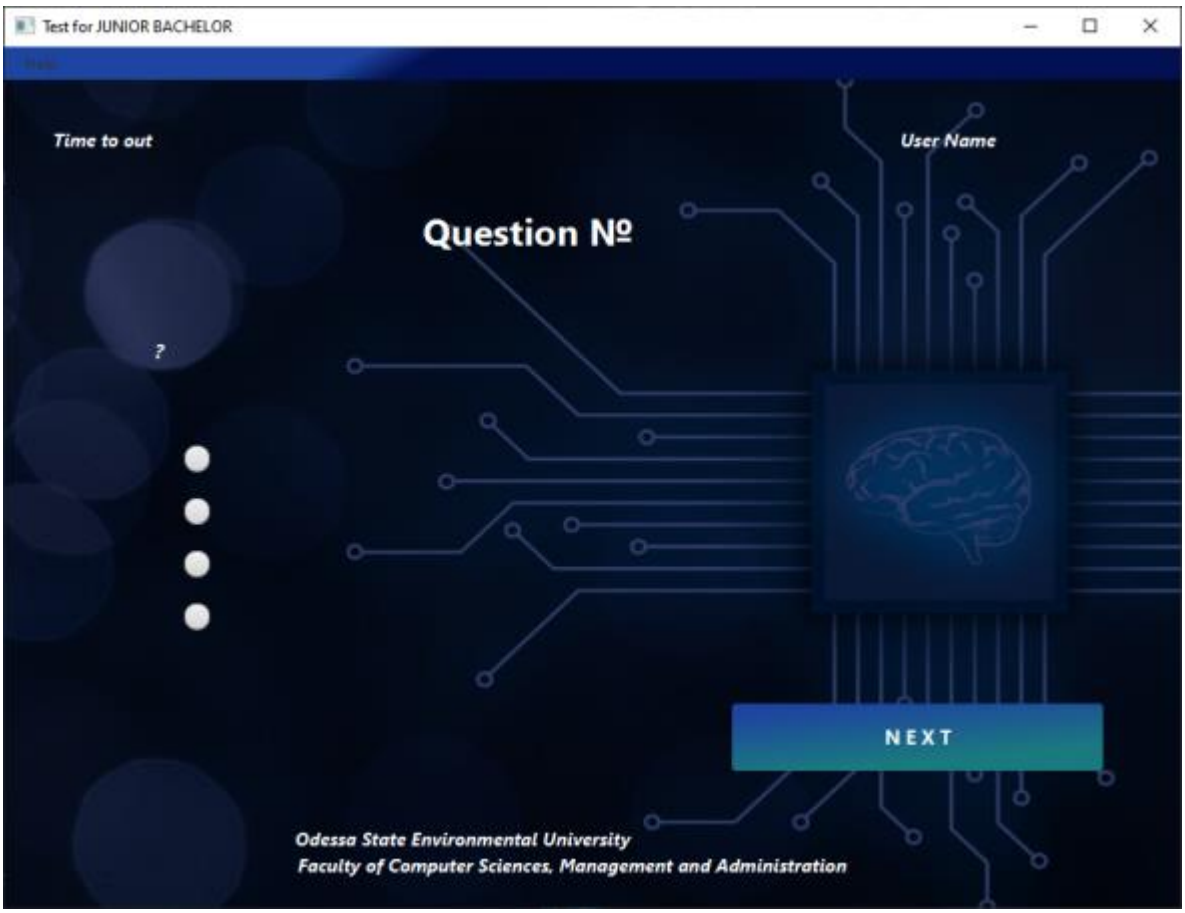

Рисунок 14 – Шаблон який використовується для формування сторінки з питанням тесту

ВИСНОВКИ

Створення БД є корисним у різних ситуаціях і для різних потреб. Нижче представлені ситуації, коли створення БД може бути особливо корисним:

- при зберігання великого обсягу даних;
- при отриманні доступу до даних із різних джерел;
- при швидкій та ефективній обробці даних;
- при організації безпеки та захисті даних;
- при організації спільної роботи та спільного доступу до даних;
- при отриманні інформації щодо аналітики та звітності.

У разі необхідності зберігати великий обсяг даних, БД надає ефективний спосіб зберігання та організації великих обсягів даних. Це особливо важливо, коли працювати необхідно з великими обсягами інформації, яку слід зберігати та обробляти. Коли потрібен доступ до даних із різних джерел, БД дозволяє централізовано зберігати дані з різних джерел та надає зручний доступ до них. Це особливо важливо, коли слід працювати з даними різних систем або джерел, які потрібно поєднати та аналізувати. Коли потрібна швидка та ефективна обробка даних, БД дозволяють виконувати операції з даними, такі як пошук, сортування, фільтрація, агрегація, швидко та ефективно. Це в свою чергу важливо, коли необхідно обробляти великі обсяги даних чи проводити складні аналітичні операції. У разі необхідності безпеки та захисту даних, БД дозволяють встановлювати заходи безпеки, такі як шифрування, автентифікація, авторизація та контроль доступу, що допомагає забезпечити захист інформації від несанкціонованого доступу. Коли потрібна спільна робота та спільний доступ до даних, БД дозволяє кільком користувачам одночасно працювати з даними, спільно колекціонувати та оновлювати інформацію. Це важливо, коли необхідно забезпечити спільну роботу команди чи розподілених користувачів над спільними даними. Якщо потрібна аналітика та звітність – БД надають можливості для аналізу даних та створення звітів, що допомагає приймати обґрунтовані рішення на основі наявної інформації[5].

Загалом створення бази даних стає корисним, коли існують значні обсяги даних, є потреба централізованого зберігання та управління інформацією, а також ефективного доступу до неї та забезпечення безпеки даних. Якщо є значна кількість даних, яку потрібно організувати і зберігати, база даних надає зручний спосіб структурування та управління цими даними. Вона дозволяє зберігати дані у відносно легкодоступному форматі і забезпечує швидкий доступ до них. Якщо багато користувачів або різні програми мають одночасний доступ до даних, база даних може служити централізованим джерелом для зберігання та обміну цими даними. Вона дозволяє керувати правами доступу до даних та забезпечує консистентність даних при одночасних операціях. База даних надає можливість встановлювати правила цілісності, які допомагають гарантувати правильність та консистентність даних. Наприклад, можна встановити обмеження унікальності значень або залежності між даними для запобігання некоректним або суперечливим даним. Також, база даних дозволяє виконувати швидкий та ефективний пошук, сортування та фільтрацію даних за допомогою запитів. Вона також підтримує індексацію даних, що сприяє прискоренню виконання операцій над даними.

Оптимізація процесу створення білетів з використанням бази даних може принести кілька переваг:

- автоматизація;
- гнучкість та масштабованість;
- централізоване зберігання даних;
- поліпшення продуктивності;
- аналіз та звітність.

Використання бази даних дозволяє автоматизувати процес створення білетів. Можна розробити відповідну структуру бази даних, що включає таблиці для питань, варіантів відповідей, ключів та зв'язку між ними. За допомогою програмування та запитів SQL можна створювати білети автоматично на основі цих даних, що суттєво прискорює процес та знижує ймовірність помилок. База даних дозволяє легко додавати нові питання та варіанти відпові-

дей, а також модифікувати існуючі. Це дозволить зберігати великий обсяг інформації, а також легко додавати та змінювати запитання та відповіді без необхідності зміни всієї структури білету. Що в свою чергу дає гнучкість в адаптації білетів під різні вимоги та оновлення контенту за необхідності. База даних надає централізоване сховище для всіх питань, відповідей та пов'язаної інформації, що забезпечує легкий доступ та керування даними. Можна зберігати та організовувати дані таким чином, щоб вони були легко доступні для генерації білетів та аналізу результатів. Використання бази даних може сприяти підвищенню продуктивності процесу створення білетів. Запити до бази даних можуть виконуватися ефективно та швидко, особливо якщо база даних оптимізована та індексована правильним чином. Це дозволяє генерувати білети швидко навіть за великого обсягу даних. Завдяки зберіганню даних у базі даних, можна проводити аналіз результатів тестування, створювати звіти та отримувати статистику про продуктивність студентів, популярність питань та інші метрики. Це може бути корисним для оцінки ефективності білетів, виявлення слабких місць та оптимізації процесу навчання.

Слід зазначити, що конкретні переваги можуть залежати від конкретного випадку використання та специфічних вимог проєкту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хмарні обчислення та захист даних у хмарі: як захистити дані у хмарних середовищах. URL: https://ts2.space/uk/%D1%85%D0%BC%D0%B0%D1%80%D0%BD%D1%96-%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F-%D1%82%D0%B0-%D0%B7%D0%B0%D1%85%D0%B8%D1%81%D1%82-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-%D1%83-%D1%85%D0%BC/ (дата звернення: 02.05.2023)
2. Методичні рекомендації щодо організації самостійної та індивідуальної роботи з навчальної дисципліни Сучасні інформаційні технології. URL: https://dduvs.in.ua/wp-content/uploads/files/Structure/library/student/nmm/0929/4.4.pdf (дата звернення: 12.04.2023)
3. Аутентифікація і авторизація: що це і в чому відмінність. URL: https://qagroup.com.ua/publications/autentyfikatciia-i-avtoryzatciia/ (дата звернення: 23.04.2023)
4. MySQL. URL: https://www.mysql.com/ (дата звернення: 23.05.2023)
5. Автоматизація за допомогою ERP Odoo: виклики та можливості. URL: https://www.buh24.com.ua/avtomatyzacziya-za-dopomogoyu-erp-odoo-vyklyky-ta-mozhlyvosti/ (дата звернення: 03.04.2023)