ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРАКТИЧНОГО НАВЧАННЯ З ТЕМИ «СТВОРЕННЯ БАЗИ ДАНИХ» ДИСЦИПЛІНИ «РОЗПОДІЛЕНІ ІНФОРМАЦІЙНО-АНАЛІТИЧНІ СИСТЕМИ»

зі спеціальності 122 Комп'ютерні науки освітня
програма «Комп'ютерні науки» ступеня магістра

Виконавець роботи Писанський Олег Русланович
_____«____»_____ 202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Ольховська Олена Володимирівна
_____«____»_____ 202_ р.
(підпис)

Рецензент

ПОЛТАВА 2023 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
C#	Це програмна мова, спрямована на об'єкти, і створена фірмою Microsoft. Вона була представлена у 2000 році як складова розробленого нею програмного фреймворку Microsoft .NET Framework.
Label	Елемент Label використовується для вставки тексту в стандартну форму, такий як заголовки, написи до полів і інша інформація.
Button	Кнопка управління
.cs	У цих файлах міститься код на мові C#, який пізніше може бути скомпільований у виконуваний файл або бібліотеку класів.
CheckBox	CheckBox в C# представляє собою елемент управління, який надає можливість користувачеві встановлювати (або скасовувати) прапорець. Використовується для отримання булевого значення (вибрано/не вибрано). Коли CheckBox встановлено, він може відображатися як квадрат з галочкою всередині, якщо користувач обрав його.

ВСТУП

В наш час величезна кількість фірм використовують персональні комп'ютери для збереження та обробки будь-якого виду інформації. Ця інформація міститься у базах даних. Бази даних відіграють важливу роль у світі технологій, що розвивається. Все, з чим ми щодня взаємодіємо в житті, очевидно, зафіксовано в якійсь базі. Робота з базами даних є найважливішою навичкою в роботі з комп'ютером, а фахівці цієї галузі стають все більш затребуваними. Головні ідеї нинішньої інформаційної методики базуються на поданні, відповідно до чого інформація повинна бути утворена в бази даних із завданням відображення світу, що динамічно змінюється, і задоволення всіх потреб в інформації у користувачів. Бази даних формуються та працюють під управлінням спеціальних програмних засобів, які називаються системами управління базами даних.

База даних, яка представлена в об'єктивній формі, це сукупність таких матеріалів: статей, рахунків, нормативних актів, судових рішень або інших подібних матеріалів, зібрані разом таким чином, щоб ці матеріали могли бути знайдені та оброблені за допомогою електронної обчислювальної машини.

База даних - це організована структура, що призначена для зберігання інформації. У той час, коли відбувався розвиток терміну баз даних, у них зберігалась виключно інформація, проте вже в наші дні багато систем управління базами даних дозволяє розміщувати у своїх структурах і дані, і програмний код, за допомогою якого відбувається зв'язок з користувачами або з іншими програмно-апаратними комплексами. При цьому дані повинні не суперечити один одному, цілісні та не надмірні. База даних створюється для збереження та безпосереднього доступу до інформації, що містить відомості про предметну область. Ступінь конкретизації даних обумовлюється групою факторів. Насамперед, метою використання

інформації з баз даних та складністю інформаційних процесів, що існують у межах предметної області у конкретних умовах.

Система управління базами даних - це програмний механізм, призначений для запису, пошуку, сортування, обробки та друку інформації, що міститься у базі даних.

У комп'ютері дані бази даних представляється як таблиці, схожої на електронну таблицю. Назви стовпців, що представляють заголовки таблиці, називають іменами полів, а стовпці - полями. Дані, що знаходяться на полях, називають значеннями полів.

Самі бази даних - це сховища безлічі систематизованої інформації, із якими виробляються такі операції: зміна, копіювання, видалення, додавання, упорядкування. Накопичення збереженого обсягу інформації, зростання групи користувачів інформаційних систем є джерелом до розвитку найкомфортніших в інтерфейсі і щодо легких розуміння табличних систем управління базами даних. Створення доступу до інформації бази даних відразу декількох користувачів одночасного, що часто перебувають на далекій відстані від місця зберігання баз даних, а також один від одного, тому і створені розраховані на багато користувачів мережевої версії баз даних сформовані на табличній структурі. Вони вирішуються проблеми притаманні паралельних процесів, правильності даних, і навіть отримання несанкціонованого входу.

За останні роки йде спостереження за напрямком до ускладнення структури даних. Прості типи інформації, що подаються у вигляді текстових рядків та чисел, не втративши своєї важливості, доповнюються сьогодні великою кількістю документів, які використовують засоби мультимедіа, образів графіки, процедурних чи активних даних та великою кількістю інших ускладнених форм інформації. З цієї причини з'явився ряд дуже витончених систем управління базами даних, що забезпечують нові колекції даних і вміють реалізувати переваги сучасних апаратних технологій. Однією з таких систем управління базами даних називається Microsoft

Access, яка входить до складу пакету програм Microsoft Office, і є популярною табличною системою управління базами даних для персональних комп'ютерів. А також MySQL, яка є найпоширенішою повноцінною серверною системою управління базами даних. [2]

Мета кваліфікаційної роботи – проектування та програмна реалізація системи практичного навчання з теми «Створення бази даних» дисципліни «Розподілені інформаційно-аналітичні системи».

Об’єкт розробки – алгоритм та програмна реалізація програмного забезпечення системи практичного навчання з теми «Створення бази даних» дисципліни «Розподілені інформаційно-аналітичні системи».

Предмет розробки – програмна реалізація системи практичного навчання з теми «Створення бази даних» дисципліни «Розподілені інформаційно-аналітичні системи».

Новизна роботи – при розробці програмного забезпечення тренажеру використано сучасні методи розробки.

Кваліфікаційна робота складається з чотирьох розділів. У першому розділі описано постановку задачі для реалізації програмного забезпечення. У другому розділі описано про актуальність бази даних. У третьому розділі розглянуто теоретичний матеріал з теми програмного забезпечення, детально описаний алгоритм роботи та блок-схема. У четвертому розділі описано процес програмної реалізації програмного забезпечення та інструкція користувача. Обсяг пояснювальної записки: 47 стор., в т.ч. основна частина – 36 стор., джерела – 15 назв.

1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ

Однією з ключових задач кваліфікаційної роботи є проектування та програмна реалізація системи практичного навчання за темою "Створення бази даних" в межах дисципліни "Розподілені інформаційно-аналітичні системи".

Мета цієї роботи визначається низкою завдань, а саме:

1. Ознайомлення з відповідним матеріалом, що стосується теми "Бази даних".
2. Розробка алгоритму для компонентів, які будуть використовуватися в програмному забезпеченні.
3. Створення блок-схеми, яка відображає алгоритм роботи додатку.
4. Програмування окремих компонентів для додатку.
5. Оцінка та аналіз результатів розробки компонентів додатку.

Ці завдання визначають різні етапи роботи, починаючи від теоретичного вивчення матеріалу і закінчуючи оцінкою відповідності розроблених компонентів поставленим цілям.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Актуальність бази даних

Будь-яка сучасна організація неспроможна уникнути бази даних. Це навчальні заклади, банки, магазини, заводи, будь-які підприємства та державні установи. Вони використовують їх для переведення даних в електронний вигляд та об'єднання даних, а також оперативного доступу до них. Це дозволяє економити час та кошти на витрати.

Звичайно, зниження часу є лише побічним ефектом автоматизації. Найголовніше завдання розвитку інформаційних технологій у зовсім іншому - у придбанні тією чи іншою організацією виключно нових якостей, що надають їй суттєвої конкурентоспроможності. А це дорогого варте.

До того ж, зараз встановлення та управління бази даних не є таким вже й важким процесом, як це було десятиліття тому. Коли проектування та управління базами даних не були автоматизовані. Система управління базою даних дозволяє створювати базу даних, оновлюючи в ній інформацію, що зберігається, забезпечуючи оперативний доступ до неї для перегляду і пошуку інформації.

Актуальність теми полягає в тому, що в нових системах управління базами даних є функція не тільки зберігання даних у своїх структурах, проте можна і зберігати програмний код, за підтримки якого йде взаємодія з користувачем або програмно - апаратним засобом.[3]

2.2. Структури та технології баз даних

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації (т.зв. записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин : збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Структуровані БД використовують структури даних, тобто структурований опис типу фактів за допомогою схеми даних, більш відомої як модель даних. Модель даних описує об'єкти та взаємовідносини між ними. Існує декілька моделей (чи типів) баз даних, основні: плоска, ієрархічна, мережна та реляційна. Приблизно з 2000 року більше половини БД використовують реляційну модель.

До **неструктурованих** БД відносяться повнотекстові бази даних, які містять неструктуровані тексти статей чи книг у формі, що дозволяє здійснювати швидкий пошук (як наприклад вікіпедія).

Часто зустрічається характеристика БД на основі певних параметрів або необхідних вимог, наприклад:

- значна кількість даних
- незалежність даних
- відкритий доступ до даних
- підтримка транзакцій з гарантією відповідних властивостей
- гарантована відсутність збоїв
- одночасна робота з багатьма користувачами

З подальшим розвитком БД змінюються й ці вимоги та додаються нові, тому одностайності щодо повноти цієї характеристики немає.

Реалізації

Комерційні

- DB2
- Informix

- Oracle
- SQL Server

З відкритим кодом

- MySQL
- Firebird
- PostgreSQL

Інтерфейс ODBC (Open Database Connectivity) був розроблений фірмою Microsoft як відкритий інтерфейс доступу до баз даних. Він надає уніфіковані кошти взаємодії прикладної програми, називаної клієнтом (або додатком- клієнтом), із сервером - базою даних.

В основу інтерфейсу ODBC були покладені специфікація CLi-Интерфейса (Call-Level Interface), розроблена X/Open, і ISO/IEC для API баз даних, а також мова SQL (Structured Query Language) як стандарт мови доступу до баз даних.

Інтерфейс ODBC проектував для підтримки максимальної интероперабельности додатків, що забезпечує уніфікований доступ будь-якого додатка, що використовує ODBC, до різних джерел даних. Так, якщо додаток, що відповідає стандарту ODBC і SQL, спочатку розроблявся для роботи з базою даних Microsoft Access, а потім таблиці цієї бази були перенесені в базу даних Microsoft SQL Server або базу даних Oracle, те додаток зможе й далі обробляти ці дані без внесення додаткових змін.

Для взаємодії з базою даних додаток- клієнт викликає функції інтерфейсу ODBC, які реалізовані в спеціальних модулях, називаних ODBC-Драйверами. Як правило, ODBC- Драйвери - це DLL- Бібліотеки, при цьому одна DLL- Бібліотека може підтримувати трохи ODBC- Драйверів. При установці на комп'ютер будь-якого SQL- Сервера (бази даних, підтримуючої один зі стандартів мови SQL, наприклад, SQL-92) автоматично виконується реєстрація в реєстрі Windows і відповідного ODBC- Драйвера.

Архітектура ODBC

Архітектура ODBC (рис. 2.1) представлена чотирма компонентами:

- Додаток-Клієнт, що виконує виклик функцій ODBC.
- Менеджер драйверів, що завантажує й звільняє ODBC- Драйвери, які потрібні для додатків- клієнтів. Менеджер драйверів обробляє виклики ODBC- Функцій або передає їхньому драйверу.
- ODBC- Драйвер, що обробляє виклики SQL- Функцій, передаючи SQL- Серверу виконуваний SQL-оператор, а додатку- клієнтові - результат виконання викликаної функції.
- Джерело даних, обумовлений як конкретна локальна або віддалена база даних.

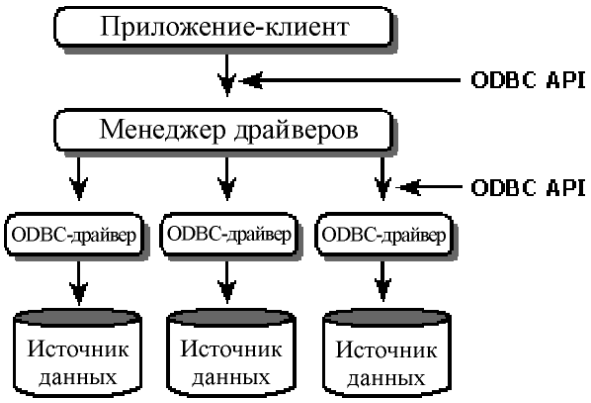

Рисунок 2.1 – Архітектура ODBC

Основне призначення менеджера драйверів - завантаження драйвера, що відповідає підключається источнику, що, даних, і інкапсуляція взаємодії з різними типами джерел даних за допомогою застосування різних ODBC-Драйверів.

ODBC-Драйвери, приймаючи виклики функцій, взаємодіють із додатком- клієнтом, виконуючи наступні завдання:

- керування комунікаційними протоколами між додатком-клієнтом і джерелом даних;
- керування запитами до СУБД;

- виконання передачі даних від додатка- клієнта в СУБД і з бази даних у додаток- клієнт;
- повернення додатку- клієнтові стандартної інформації про виконаний виклик ODBC- Функції у вигляді коду повернення;
- підтримує роботу з курсорами й управляє шипиками.

Додаток-Клієнт одночасно може встановлювати з'єднання з декількома різними джерелами даних, використовуючи різні ODBC- Драйвери, а також кілька з'єднань із тим самим джерелом даних, використовуючи той самий ODBC- Драйвер.

Функції ODBC API

Всі функції ODBC API умовно можна розділити на чотири групи:

- основні функції ODBC, що забезпечують взаємодію із джерелом даних;
- функції установки (setup DLL);
- функції інсталяції (installer DLL) ODBC і джерел даних;
- функції перетворення даних (translation DLL).

Оголошення всіх функцій і використовуваних ними типів даних утримуються в заголовних файлах. Група основних функцій ODBC API розбита на три з:

- функції ядра ODBC;
- функції 1 рівня;
- функції 2 рівні.

Кожний ODBC- Драйвер специфицируется як драйвер, що підтримує певний рівень функцій ODBC API.

Прототипи функцій ядра ODBC API перебувають у файлі Sql.h (C/C++, Visual Studio), а прототипи функцій 1 і 2 рівнів - у файлі Sqlext.h.

Застосування #define ODBCVER дозволяє вказати використовувану версію (наприклад, #define ODBCVER 0x0351). Прототипи функцій установки й інсталяції перебувають у файлі odbcinst.h.

Відкритий інтерфейс доступу до баз дані фірми Microsoft заснований на наступних стандартах:

- специфікація X/Open CAE1) (Specification "Data Management: SQL Call-Level Interface (CLI)");
- специфікація ISO2) /IEC 9075-3:1995 (E) (Call-Level Interface (SQL/CLI)).

У цей час фірма Microsoft підтримує версію 3.x ODBC API. Додатка, написані на основі специфікації X/Open і ISO CLI, будуть правильно працювати з ODBC- Драйверами версії 3.x або драйверами "погодженого стандарту" у тому випадку, якщо вони компілюються із заголовними файлами ODBC версії 3.x і линкуются з ODBC 3.x бібліотеками, а доступ до ODBC- Драйвера одержують через менеджер драйверів ODBC 3.x. Аналогічно, що й самі драйвери 3.x, написані на основі специфікації X/Open і ISO CLI, будуть правильно працювати з додатками при дотриманні цих же умов.

Драйвер ODBC 3.x завжди підтримує всі можливості, використовувані додатком "погодженого стандарту", а додаток ODBC 3, що використовує тільки можливості, надавані ISO CLI, і обов'язкові засоби, описувані X/Open CLI, завжди буде працювати із драйвером "погодженого стандарту".

На додаток до інтерфейсу, специфікованому в стандартах ISO/IEC і X/Open CLI, ODBC реалізує наступні можливості:

- витяг декількох рядків (блокова вибірка) за один виклик функції;
- зв'язування з масивом параметрів;
- підтримка закладок, включаючи вибірку за допомогою закладки, закладки змінної довжини, блокове відновлення й видалення за допомогою відзначених операцій над непослідовними рядками;
- порядкове зв'язування (row-wise binding);
- зв'язування зі зсувом (binding offsets);

- підтримка пакетів SQL- Операторів як у збережених процедурах, так і у вигляді послідовності окремих SQL- Операторів, виконуваних при виклику функцій SQLExecute і SQLExecDirect;
- визначення точного або приблизного числа рядків курсору;
- застосування операції присохне відновлення й видалення й пакетних видалень і відновлень із використанням функції SQLSetPos;
- підтримка функцій каталогу, що дозволяють одержувати інформацію зі схеми бази даних (системних таблиць);
- бібліотеки перетворення для кодових сторінок;
- асинхронне виконання;
- підтримка збережених процедур, включаючи escape-послідовності, механізм зв'язування вихідних параметрів, функції каталогу;
- більше просунуті можливості з'єднання, що включають підтримку атрибутів з'єднання й перегляду атрибутів.

Створення джерела даних

Джерело даних DSN, використовуваний функціями ODBC API, спочатку повинен бути створений. Це можна виконати як програмно - викликавши функцію ODBC API, так і интерактивно - використовуючи утиліту ODBC (залежно від версії Windows, розташовану на панелі керування або адміністрування).

DLL-Бібліотека ODBCCP32.DLL надає функції ODBC API ConfigDSN і SQLConfigDataSource, що дозволяють виконувати реєстрацію нових джерел даних або видаляти інформацію про джерела даних з реєстру Windows (і з файлу ODBC.ini).

Функція ConfigDSN дозволяє додавати, змінювати або видаляти джерела даних і має наступний формальний опис:

BOOL ConfigDSN(

HWND hwndParent,

WORD fRequest,

LPCSTR lpszDriver,

LPCSTR lpszAttributes);

Для використання в середовищі Visual Studio функцій ConfigDSN і SQLConfigDataSource варто підключити заголовний файл odbcinst.h.

Параметр hwndParent визначає дескриптор вікна або NULL. Якщо дескриптор не зазначений, то при виконанні даної функції вікно із пропозицією уточнити параметри не відображається. Параметр fRequest указує тип запиту, що задається однієї з наступних констант:

ODBC_ADD_DSN - додавання нового джерела даних;

ODBC_CONFI_DSN - зміна існуючого джерела даних;

ODBC_REMOVE_DSN - видалення існуючого джерела даних.

Параметр lpszDriver містить опис драйвера, а параметр lpszAttributes - список атрибутів у формі "ключове слово=значення" (наприклад: DSN=MyDB\0UI=U1\0PWD=P1\0DATABASE=DB1\0\0). Список атрибутів завершується двома null-байтами

При успішному завершенні функція повертає значення TRUE, у противному випадку викликом функції SQLinstallerError можна одержати один з наступних кодів помилки:

ODBC_ERROR_INVALI_HWND - помилка у вказівці дескриптора вікна;

ODBC_ERROR_INVALI_KEYWORD_VALUE - параметр lpszAttributes містить помилки;

ODBC_ERROR_INVALI_NAME - параметр lpszDriver не знайдений у системі;

ODBC_ERROR_INVALI_REQUEST_TYPE - параметр fRequest містить неприпустиме значення;

ODBC_ERROR_REQUEST_FAILED - не можна виконати дія, зазначена параметром fRequest ;

ODBC_ERROR_DRIVER_SPECIFI - помилка конкретного драйвера.

Для запису інформації про джерело даних у секцію [ODBC Data Sources] секції ODBC.INI реєстру Windows функція ConfigDSN викликає функцію SQLWriteDSNToini, а для видалення - функцію SQLRemoveDSNFromini.

Функція SQLConfigDataSource має наступний формальний опис:

```
BOOL SQLConfigDataSource(
 HWND hwndParent,
 WORD fRequest,
 LPCSTR lpszDriver,
 LPCSTR lpszAttributes);
```

Параметри функції SQLConfigDataSource аналогічні параметрам функції ConfigDSN, при цьому параметр fRequest може приймати наступні значення:

ODBC_ADD_DSN - додавання нового користувальницького DSN;

ODBC_CONFI_DSN - зміна існуючого користувальницького DSN;

ODBC_REMOVE_DSN - видалення існуючого користувальницького DSN;

ODBC_ADD_SYS_DSN - додавання нового системного DSN;

ODBC_CONFI_SYS_DSN - зміна існуючого системного DSN;

ODBC_REMOVE_SYS_DSN - видалення існуючого системного DSN.

Функція ConfigDSN ставиться до групи функцій установки DLL (setup DLL), а функція SQLConfigDataSource - до групи функцій інсталяції DLL (Installer DLL).

При виконанні функція SQLConfigDataSource використовує значення параметра lpszDriver для одержання із системної інформації повного шляху до Setup DLL конкретного драйвера, завантажує цю DLL і викликає функцію ConfigDSN, передаючи їй свій список параметрів (значення параметра fRequest перетвориться до значення, прийнятому функцією ConfigDSN). Перед викликом ConfigDSN, залежно від типу оброблюваного

DSN, установлюється режим USERDSN_ONLY (користувальницький DSN) або SYSTEMDSN_ONLY (системний DSN), а перед завершенням виконання функція SQLConfigDataSource повертає режим BOTHDSN.

Як ми вже зазначали, всі функції ODBC API умовно можна розділити на чотири групи:

- основні функції ODBC, що забезпечують взаємодію із джерелом даних;
- функції установки (setup DLL);
- функції інсталяції (installer DLL) ODBC і джерел даних;
- функції перетворення даних (translation DLL), викликувані при передачі даних від драйвера до джерела даних або назад.

У наступній таблиці представлений список основних функцій ODBC API і їхній рівень відповідності стандартам (у стовпці "Відповідність" показана відповідність стандартам і зазначена версія ODBC, починаючи з якої дана функція доступна):

Якщо додаток використовує функції ODBC 2.x з менеджером драйверів ODBC 3.x, то менеджер драйверів підмінює функцію й передає драйверу ODBC 3.x виклик функції у відповідності з наступною таблицею (таблиця 1.1).

Таблиця 1.1 - список основних функцій ODBC API

Функції ODBC 2.x	Функції ODBC 3.x
SQLAllocConnect	SQLAllocHandle
SQLAllocEnv	SQLAllocHandle
SQLAllocStmt	SQLAllocHandle
SQLBindParam (для стандарту X/Open і ISO)	SQLBindParameter
SQLColAttributes	SQLColAttribute
SQLError	SQLGetDiagRec

SQLFreeConnect	SQLFreeHandle
SQLFreeEnv	SQLFreeHandle
SQLFreeStmt	SQLFreeHandle
SQLGetConnectOption	SQLGetConnectAttr
SQLGetStmtOption	SQLGetStmtAttr
SQLParamOptions	SQLSetStmtAttr
SQLSetConnectOption	SQLSetConnectAttr
SQLSetParam [функція ODBC 1.0]	SQLBindParameter
SQLSetScrollOption	SQLSetStmtAttr
SQLSetStmtOption	SQLSetStmtAttr
SQLTransact	SQLEndTran

Перед використанням функцій ODBC API додаток- клієнт створює дескриптор (ідентифікатор) оточення, що визначає глобальний контекст для доступу до джерел даних. Дескриптор оточення надає доступ до різної інформації, включаючи поточні установки всіх атрибутів оточення, дескриптори з'єднань, створені для даного оточення, діагностику рівня оточення.

Дескриптор оточення визначає деяку структуру, що містить дану інформацію. Безпосередньо дескриптор оточення звичайно використовується при виклику функцій SQLDataSources і SQLDrivers і при створенні дескрипторів з'єднання.

Для додатка- клієнта, що реалізує з використанням функцій ODBC API доступ до джерела даних, досить мати один дескриптор оточення.

Створення дескриптора оточення виконується функцією SQLAllocHandle, а звільнення - функцією SQLFreeHandle.

Функція SQLAllocHandle введена у версії ODBC 3.x замість існуючих у версії ODBC 2.0 функцій SQLAllocConnect, SQLAllocEnv і SQLAllocStmt. Для того щоб додаток, що використовує функцію SQLAllocHandle, міг працювати через драйвери ODBC 2.x, менеджер драйверів версії 3.x заміняє

виклики функцій третьої версії на їхні аналоги другої версії й передає такий "відкоректований" виклик ODBC-драйверу.

Для підключення до бази даних варто створити дескриптор (ідентифікатор) з'єднання. Для одного дескриптора оточення може бути створено кілька дескрипторів з'єднання.

Для виконання SQL- Оператора створюється дескриптор (ідентифікатор) оператора.

Для одного дескриптора з'єднання може бути створено кілька дескрипторів оператора.

По специфікації ODBC для кожного додатка драйвери можуть підтримувати необмежене число дескрипторів кожного типу. Однак конкретний драйвер може накладати деякі обмеження на кількість дескрипторів.

Функція, використовувана для створення дескриптора оточення, з'єднання, оператора або додатки, має наступний формальний опис:

```
SQLRETURN SQLAllocHandle(
 SQLSMALLINT HandleType,
 SQLHANDLE InputHandle,
 SQLHANDLE * OutputHandlePtr);
```

Параметр HandleType ([Input]) указує однієї з наступних констант тип створюваного дескриптора:

SQL_HANDLE_ENV

SQL_HANDLE_DBC

SQL_HANDLE_STMT

SQL_HANDLE_DESC

Параметр InputHandle ([Input]) уизначає контекст, у який додається створюваний дескриптор. Якщо тип дескриптора SQL_HANDLE_ENV, то параметр InputHandle вказується константою SQL_NULL_HANDLE. При створенні дескриптора середовища параметр InputHandle задає дескриптор

оточення, а для створення дескриптора оператора (SQL_HANDLE_STMT) і дескриптора додатка (SQL_HANDLE_DESC) - дескриптор з'єднання.

Ідентифікатори, що визначають тип дескриптора й сам дескриптор, описані в заголовних файлах sql.h і sqltypes.h у такий спосіб:

```
/* sql.h */
#if (ODBCVER >= 0x0300)
#define SQL_HANDLE_ENV 1
#define SQL_HANDLE_DBC 2
#define SQL_HANDLE_STMT 3
#define SQL_HANDLE_DESC 4
#endif
/* sqltypes.h */
#if (ODBCVER >= 0x0300)
#if defined(WI32) || defined(_WI64)
typedef void*SQLHANDLE;
#else
typedef SQLINTEGER SQLHANDLE;
#endif  /* defined(WI32) || defined(_WI64) */
typedef SQLHANDLE SQLHENV;
typedef SQLHANDLE SQLHDBC;
typedef SQLHANDLE SQLHSTMT;
typedef SQLHANDLE SQLHDESC;
#else //ODBCVER < 0x0300
#if defined(WI32) || defined(_WI64)
typedef void*                   SQLHENV;
typedef void*                   SQLHDBC;
typedef void*                   SQLHSTMT;
#elsetypedef SQLINTEGER SQLHENV;
typedef SQLINTEGER SQLHDBC;
typedef SQLINTEGER SQLHSTMT;
```

#endif /* defined(WI32) || defined(_WI64) */

#endif /* ODBCVER >= 0x0300 */

Параметр OutputHandlePtr ([Output]) - це вказівник на буфер, у який міститься створювана для дескриптора структура даних.

Функція SQLAllocHandle може повертати наступні значення:

SQL_SUCCESS - значення, обумовлене ODBC API для вказівки успішного завершення функції;

SQL_SUCCESS_WITH_INFO - значення, обумовлене ODBC API для вказівки того, що функція виконана успішно, але з повідомним повідомленням;

SQL_INVALI_HANDLE - значення, обумовлене ODBC API для вказівки, що задано невірний дескриптор;

SQL_ERROR - значення, обумовлене ODBC API для вказівки, що при виконанні функції відбулася помилка.

Для одержання додаткової інформації про помилку виконання функції додаток- клієнт може використовувати дані з дескриптора, зазначеного параметром InputHandle.

Якщо при виконанні функції відбулася помилка (код повернення SQL_ERROR) або функція виконана, але з повідомним повідомленням (код повернення SQL_SUCCESS_WITH_INFO), то значення SQLSTATE можна одержати при виклику функції SQLGetDiagRec.

Після створення дескриптора оточення варто встановити атрибут SQL_ATTR_ODBC_VERSION. У противному випадку при спробі створити дескриптор з'єднання відбудеться помилка.

Для додатків "погодженого стандарту" під час компіляції функція SQLAllocHandle заміняється на SQLAllocHandleStd. Основна відмінність останньої полягає в тому, що при виклику цієї функції зі значенням параметра HandleType, рівним SQL_HANDLE_ENV, відбувається установка атрибута оточення SQL_ATTR_ODBC_VERSION, рівним

SQL_OV_ODBC3 (тому що додатка "погодженого стандарту" завжди є додатками ODBC 3.x і не вимагають реєстрації версії додатка). [4]

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Застосування баз даних

Розглянемо таку проблему, як складання та ведення журналу учнів. Відбувається зіткнення не з малим обсягом однакових даних про учнів (місце проживання, персональні дані, відомості про батьків та інше) та процес навчання (успішність, контрольні, уроки, заходи та інше). Для автоматизації цього завдання застосування алгоритмічних мов не підходить. Власне, для цього і призначена система управління базами даних. Ця система не приєднується до вирішення одних певних завдань. Вони автоматизовані шаблонні операції, необхідних роботи з базами даних, оскільки удосконалення триває, то наступної версії чи новому варіанті системи управління базами даних реалізовано дедалі більше подібних операцій. Вирішення проблем оптимізації за допомогою систем управління базами даних призводить до створення інформаційних систем.

Інформаційна система - система, призначена для пошуку, зберігання та обробки інформації, та відповідних організаційних ресурсів (технічні, фінансові, людські тощо), які забезпечують та поширюють інформацію.

Інформаційні системи виникли ще у 60-х роках минулого століття у військовій індустрії та бізнесі, де було накопичено великі обсяги корисних відомостей. Спочатку інформаційні системи призначалися лише працювати з інформацією фактичного виду - текстові чи числові характеристики об'єктів. Потім, у міру розвитку технічного оснащення комп'ютерів, стало можливим обробка текстових даних природною мовою.

Інформаційна система служить для оперативного забезпечення певної групи осіб необхідними даними, тобто задоволення будь-яких інформаційних потреб у межах обраної предметної області, у своїй результатом роботи інформаційних систем є інформаційна продукція – текстові документи, масиви інформації, бази даних. По області застосування розрізняють дві основні групи інформаційних систем:

- інформаційно-пошукові системи;
- системи обробки даних.

Інформаційно-пошукові системи спрямовані на отримання даних, що зберігаються, що задовольняють будь-якому запиту. У цьому користувача не цікавить результати обробки даних, скільки шукана інформація (наприклад, яку стипендію отримає студент Петров наступного семестру).

Використання користувачами систем обробки інформації в більшості випадків є причиною зміни даних. Вивід даних вірно буде сказати відсутній або представляє результат програмної обробки інформації, що зберігається, але не саму інформацію.

Обробка даних - спеціальний клас розв'язуваних на персональному комп'ютері завдань, пов'язаних з відбором, групуванням, зберіганням, сортуванням та введенням записів інформації однорідної структури. До завдань класу відносяться:

- підрахунок вироблених деталей на заводі;
- зарахування коштів;
- управління персоналом, бухгалтерським обліком, зв'язком тощо.

Існують фактографічні інформаційні системи ефективної обробки даних, що передбачають негайне обслуговування абсолютно звичайних запитів від значної кількості споживачів, і фактографічні інформаційні системи дослідницької обробки, орієнтованих на реалізацію різноманітних запитів, які вимагають виконання статистичної обробки історичної (накопиченої за довгий час) інформації, та передбачення розвитку цих процедур. Тому інформаційні системи застосовують у наступних сферах:

- створення сховища для інформації;
- концепція оцінки інформації;
- налагодженість затвердження постанови;
- мобільні та локальні бази даних;
- географічні бази даних;

- мультимедіа бази даних;
- розподілені інформаційні системи;
- бази даних для всесвітньої мережі.

Удосконалення інформаційних технологій супроводжує двома дуже цікавими напрямками в тому, що стосується термінології. По-перше, безперервна зміна імен для тих самих речей (хоч і технології розвиваються, але швидкість їх набагато нижче, ніж імен). По-друге, використання старих термінів для понять, які вже зараз мають зовсім інший сенс. Власне, остання обставина застосовується саме до системи управління баз даних.

Останнім часом спостерігається така обстановка, коли системи управління базами даних перетворюється із виключно внутрішнього технологічного доповнення до прикладних програм на самостійний продукт, який будує програми для споживачів.

Слід також виділити модифікацію змісту платформи Microsoft. Зазвичай мається на увазі операційна система Windows. Але у застосуванні до серверної платформи часто зустрічається: SQL Server + Windows Server.

Згодом склалося так, що системи управління базами даних спрямовані на вирішення завдань, пов'язаних насамперед із транзакційною обробкою структурованої інформації. Абсолютно, найкращим перевіреним часом рішенням залишається таблична модель системи управління базами даних. Але останнім часом сфера застосування бази даних розширювалася. На перший погляд необхідно керувати широким набором форматів даних, переходячи до вирішення загальних проблем управління корпоративною інформацією. Але якщо з іншого боку, саме системи управління базами даних беруть він основні функції інтеграції інформації та додатків корпоративних систем. Цим пояснюється зацікавленість до обговорення архітектурних принципів та реалізації можливостей баз даних різних моделей: XML, постріляційних, об'єктно-реляційних.

Роблячи класифікацію існуючих сфер застосування баз даних, і навіть якщо дати оцінку перспективи їх еволюції у світі, то вийде приблизний

перелік особливо визнаних видів, популярних і застосування у всіх сферах застосування баз даних. Цей перелік буде виглядати так:

- документальні та документографічні, що застосовуються в будь-яких базах органів влади;
- бази даних щодо продукції промисловості, сільського господарства та будівництва;
- бази даних за статичною, кредитно-фінансовою та зовнішньоторговельною інформацією;
- фактографічні бази соціальних даних, які включають відомості про соціальне середовище та населення;
- бази даних систем транспорту;
- довідкові дані: енциклопедії, довідники, адреси та телефони організацій, розклади;
- ресурсні бази даних, що включають інформацію про такі природні ресурси: земля, вода, надра, гідрометеорологія, біоресурси, екологічна обстановка;
- фактографічні основи основних наукових досліджень;
- фактографічні бази даних у сфері мистецтва та культури;
- машинні словники різного типу та призначення.

Для вирішення проблем в економіці, які мають різновид і широкість, потрібно вдаватися до застосування програмного забезпечення системи управління базами даних. На основі цієї програми створюються інформаційні системи організацій різних рівнів (від крихітних до особливо великих). Області застосування баз даних за традицією займають ті галузі життєдіяльності людини, де вона повинна взаємодіяти з великою кількістю однотипної інформації. Першим баз даних знайшли застосування у хімії, ядерної фізики, космонавтиці та інших науках, які вимагають систематичного підходи до роботи з інформацією. Подальша еволюція комп'ютеризації і комп'ютерних показників призвело людство до того що, що бази даних опинилися у створенні практично у всіх галузях зайнятості

людини, і стали повсякденно використовуватися у різних економічних об'єктах: від сільського підприємництва до фінансових систем. Останніми нововведеннями застосування баз даних стала всесвітня мережа інтернет, яка по суті є найбільшою та великою базою даних. Відповідно, таке поширення баз даних вимагає нових програмних засобів для управління ними.

Комп'ютеризація людства виступає головною сферою автоматизування промислової та іншої установчої роботи, де обов'язкові збереження, обробка, отримання, передача та збір у єдину інформацію. Автоматизація на персональних комп'ютерах змінює стандарти переробки даних, надаючи злагоджену роботу промисловості та організацій на основі новітньої інформаційної технології.

Застосування персонального комп'ютера як механізму обробки інформації у різних галузях людської діяльності підвищує інформаційну культуру суспільства, сприяючи без ускладнень перейти до інформаційному суспільству, де інформація є найціннішим матеріалом нарівні з фінансовими, енергетичними та іншими ресурсами. У потрібний час отримана, правильно оброблена та чітко подана інформація часто збільшує ефективність прийнятих рішень і, отже, їх результат.

Будь-яка інформаційна система, залежно від призначення, має справу з тією чи іншою частиною конкретного світу, яка називається предметною областю. Вивчення предметної галузі вважається початковим етапом розробки кожної інформаційної системи. Тобто цьому етапі встановлюються потреби у інформації всієї аудиторії споживачів майбутньої системи, які, визначають вміст її бази даних. Предметна область певної інформаційної системи сприймається як деяка сукупність реальних об'єктів, які цікаві її користувачів. Таким прикладом є персональні комп'ютери, програмні засоби та їх споживачі. Всі ці об'єкти мають певні властивості. Наприклад, персональний комп'ютер має такі характеристики: найменування фірми-виробника, номер моделі, потужністю процесора,

ємністю жорсткого диска та оперативного запам'ятовуючого пристрою, параметрами графічної карти тощо.

Інформаційний об'єкт – це опис деякої сутності предметної області, реального об'єкта, процедури чи події. Інформаційний об'єкт створюється групою логічно пов'язаних атрибутів, які є якісні та чисельні характеристики сутності.

Між об'єктами предметної області можуть бути зв'язку, які мають різний зміст за змістом. Ці зв'язки можуть бути обов'язковими чи факультативними.

Якщо об'єкт виявляється за необхідності пов'язаним з будь-яким об'єктом предметної області, між цими двома об'єктами існує обов'язковий зв'язок. Інакше зв'язок є необов'язковим. Наприклад, обов'язковий зв'язок замінює існуючу між двома об'єктами "працівник" і "посада" в предметній галузі кадрових інформаційних систем, з цього випливає, що кожен найнятий в організацію співробітник зараховується на посаду і не може бути співробітника, який не займає жодної посади. У цей час зв'язок заміщає між типами об'єктів " співробітник " і " посада " є необов'язковою, оскільки може бути вільні місця працювати. Структура предметної області характеризується безліччю об'єктів предметної області та зв'язків між ними. Безліч об'єктів предметної області, значення властивостей об'єктів та зв'язку між ними можуть змінюватись у часі. Зміни можуть зводитись до появи нових або виключення з розгляду деяких існуючих об'єктів у предметній області, встановлення нових або руйнування існуючих зв'язків між ними. Отже, з кожним моментом можна порівняти деякий стан предметної області.

Інформаційно-логічна модель – це безліч інформаційних об'єктів предметної галузі та взаємозв'язків між ними. Процедура освіти інформаційної моделі починається із призначення концептуальних потреб майбутніх користувачів бази даних. Потреби окремих користувачів інтегруються в єдиному загальному уявленні, яке називається концептуальною моделлю предметної галузі. Розглянемо її (рис.3.1)

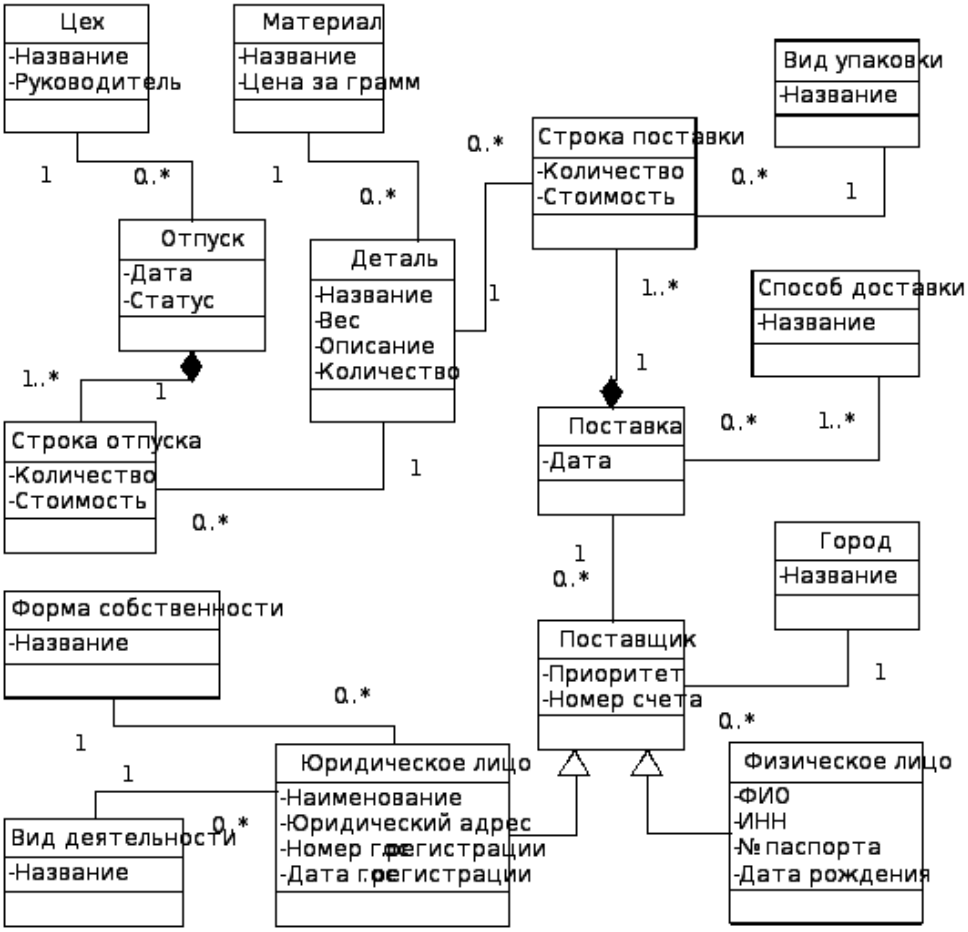

Рисунок 3.1 - концептуальна модель предметної галузі.

Ця модель являє собою предметну область в образі з'єднаних між собою об'єктів без позначення способу їх зберігання.

Концептуальна модель є об'єднаним списком бажань всіх користувачів до бази даних предметної області. Тому всі старання програміста мають бути спрямовані здебільшого на структуризацію даних, що належать майбутнім споживачам бази даних та показ зв'язків між ними.

Взаємозв'язки між об'єктами, відображені в концептуальній моделі можуть бути з часом нереалізованими засобами обраної системи управління бази даних, вимагаючи її зміни. Логічна модель - модифікація концептуальної моделі, реалізована конкретної системою управління базами даних.

Логічна модель, яка відображає логічні зв'язки між властивостями об'єктів незалежно від їх вмісту та області зберігання, може бути реляційною, ієрархічною чи мережевою.

Зовнішньою моделлю користувачів називають такі різні підмножини її логічної моделі, які відповідають різним користувачам в інформаційній моделі.

З цього виходить, що зовнішня модель користувача є відображенням його концептуальних потреб у логічній моделі і відповідає тому уявленню, з якого цей користувач черпає про предметну область на основі логічної моделі. Тому, наскільки ідеально сконструйована зовнішня модель, настільки повно і точно відображає предметну область і настільки добре виконує роботу автоматизована система управління цією предметною областю.

Інформаційні дані кожного користувача в базі даних повинні бути незалежними від інших користувачів, тобто не повинні впливати на існуючі зовнішні моделі і бути змінені ними. Такий порядок відбиває базовий рівень незалежності даних. З іншого боку, зовнішні моделі користувачів ніяк не пов'язані з типом фізичної пам'яті, в якій зберігатимуться дані, та з фізичними методами доступу до цієї інформації. Цей порядок відбиває другий рівень незалежності даних. [5]

3.2 . Алгоритм роботи системи практичного навчання

На екран виводиться стартове вікно з темою тренажера, її виконавця та керівника, нижче кнопка про початок проходження тренажера.

По черзі виводяться питання тренажера, при цьому користувач не може перейти наступне запитання без вибору відповіді, а лише вийти з нього.

У випадку, коли користувач вибирає відповідь, розблокується кнопка для наступного питання.

При проходженні всіх 6-ти питань, наприкінці тренажера виводиться результат з кількістю правильних відповідей, а також перелік питань, на які користувач відповів неправильно.

Питання 1/6. Основними поняттями ієрархічної структури є …

а) Рівень, вузол, зв'язок;

б) Відношення, атрибут, кортеж;

в) Таблиця, стовпець, рядок.

Питання 2/6. Для таблиці реляційної бази даних є хибним твердження, що

а) Кожен стовпець таблиці має унікальне ім'я;

б) Всі стовпці таблиці містять однорідні за типом дані;

в) Кожен запис у таблиці містить однорідні за типом дані.

Питання 3/6. Основними об'єктами СУБД MS Access є …

а) Таблиця, форма, звіт, запит;

б) Таблиця, поле, запис, ключ;

в) Схема даних, ключ, шаблон, звіт.

Питання 4/6. Що являє собою СУБД?

а) Програми для забезпечення взаємодії користувача з даними;

б) Комплекс програм, що забезпечує взаємодію користувача з базою даних;

в) Програми для взаємодії користувачів з операційною системою.

Питання 5/6. Що таке СУБД?

а) Скорочення від слів Система Управління Базами Даних;

б) Скорочення від слів Структура Управління Базами Даних;

в) Скорочення від слів Система Управління Базисами Даних.

Питання 6/6. Коли місце збереження інформації стає базою даних?

а) Якщо забезпечена секретність даних;

б) Якщо завідуючий базою починає виконувати свої обов’язки;

в) Якщо сховище даних відповідає визначенню бази даних.

3.3. Блок-схема алгоритму

Блок-схема - це схематичне уявлення процесу, системи чи комп'ютерного алгоритму. Блок-схеми часто застосовуються у різних сферах діяльності, щоб документувати, вивчати, планувати, удосконалювати та пояснювати складні процеси за допомогою простих логічних діаграм. Для побудови блок-схем застосовуються прямокутники, овали, ромби та деякі інші фігури (для позначення конкретних операцій), а також сполучні стрілки, які вказують послідовність кроків або напрямок процесу. Розглянемо блок схему алгоритму системи практичного навчання (рис. 3.2)

Рисунок 3.2 – блок схема алгоритму

3.4. Фізична структура алгоритму

Нижче представлено фізичну структуру алгоритму (рис.3.3)

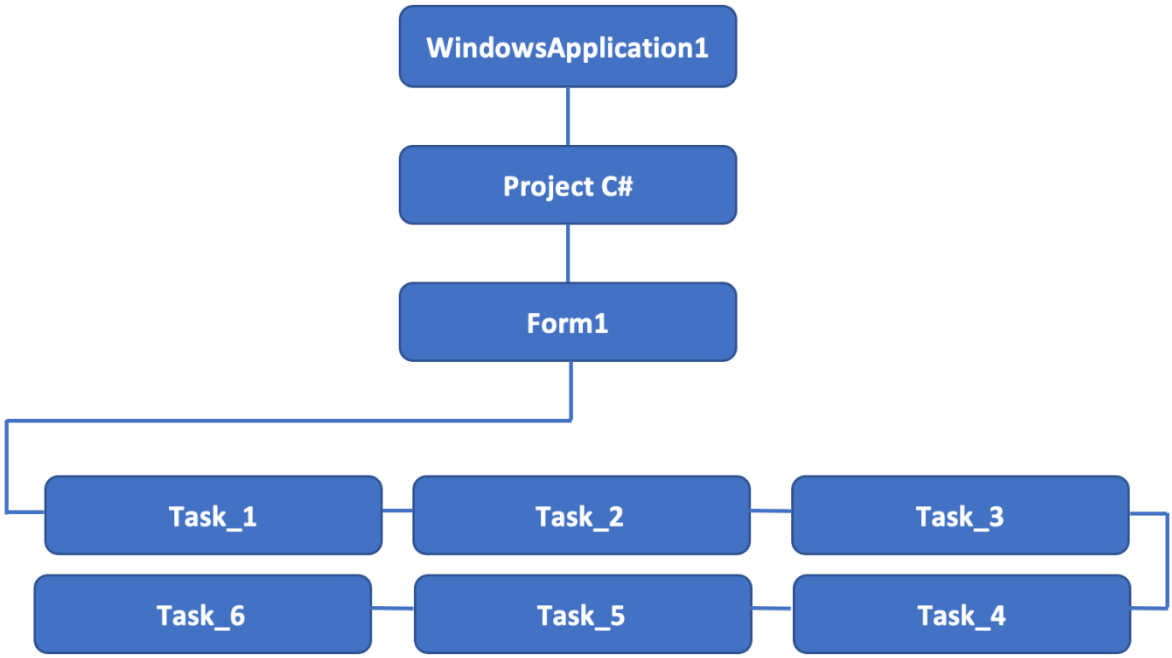

Рисунок 3.3 – Фізична структура алгоритму

3.5. Логічна структура алгоритму

Нижче представлено логічну структуру алгоритму (рис.3.4)

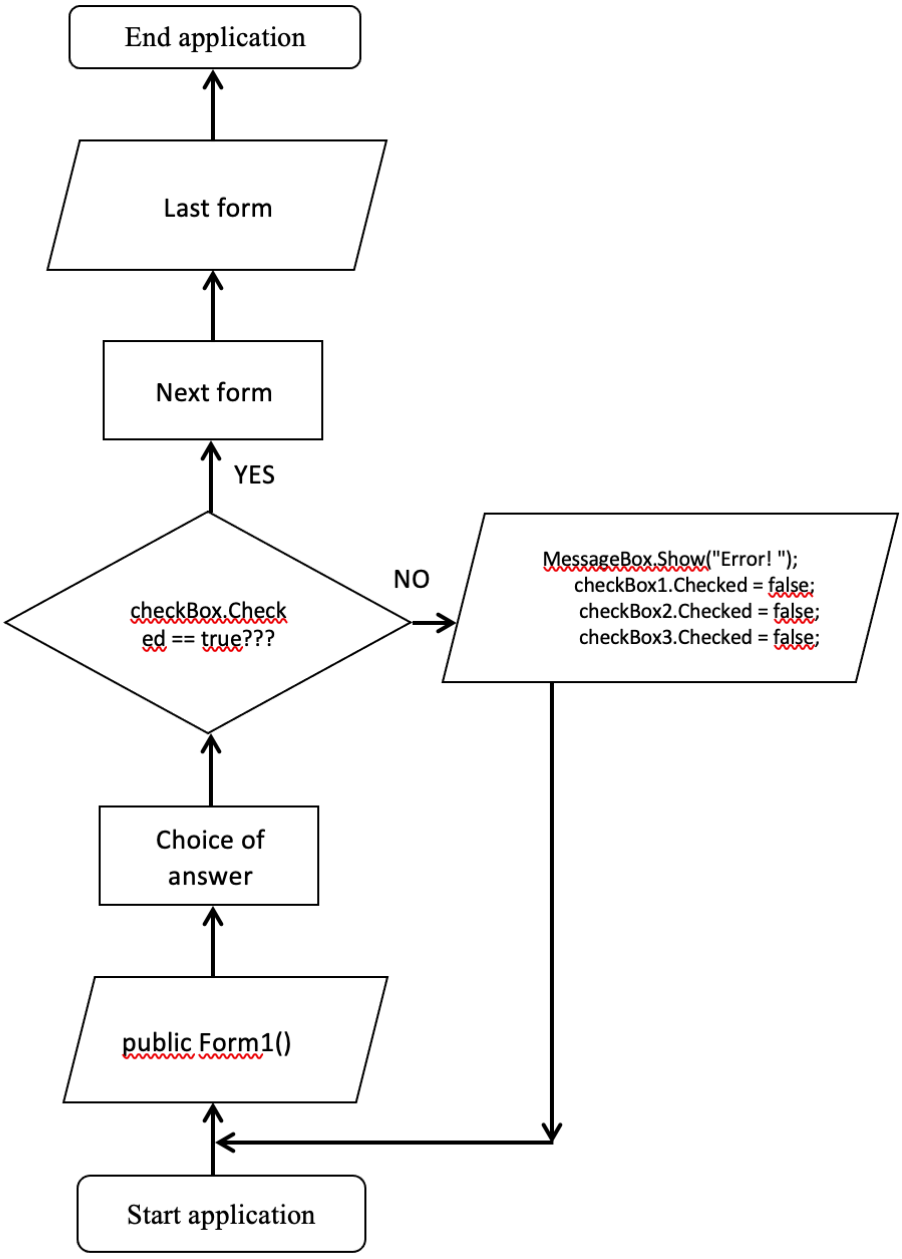

Рисунок 3.4 – логічна структура алгоритму

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів

C# широко використовується для розробки різноманітних застосунків, включаючи десктопні програми, веб-застосунки та мобільні додатки з використанням платформи Xamarin.

Інтеграція з .NET Framework:

C# є основною мовою програмування для платформи .NET Framework. Це забезпечує велику інтеграцію з іншими мовами, бібліотеками та сервісами, що полегшує розробку та підтримку програм.

Безпека та Ефективність:

C# має вбудовані механізми безпеки, такі як контроль типів та керування пам'яттю, що сприяє уникненню багатьох типових помилок під час розробки.

Об'єктно-орієнтоване програмування:

Мова підтримує об'єктно-орієнтований підхід, що сприяє легкості управління кодом, його повторному використанню та розширенню.

Інтегрована Розробка:

Середовище розробки Visual Studio надає потужні інструменти для створення, налагодження та управління проектами на C#. Це спрощує роботу розробників та підвищує продуктивність.

Підтримка Сучасних Технологій:

C# підтримує нові технології та парадигми програмування, такі як асинхронне програмування та LINQ, що дозволяє розробникам писати більш зручний та продуктивний код.

Спільнота та Документація:

Мова має активну спільноту розробників та обширну документацію, що полегшує вирішення проблем, обмін досвідом та пошук рішень.

4.2. Опис програмної реалізації

Для початку роботи необхідно створити проект в середовищі Visual Studio [7, 8, 9] та задати в ньому певну ієрархію. (рис. 4.1)

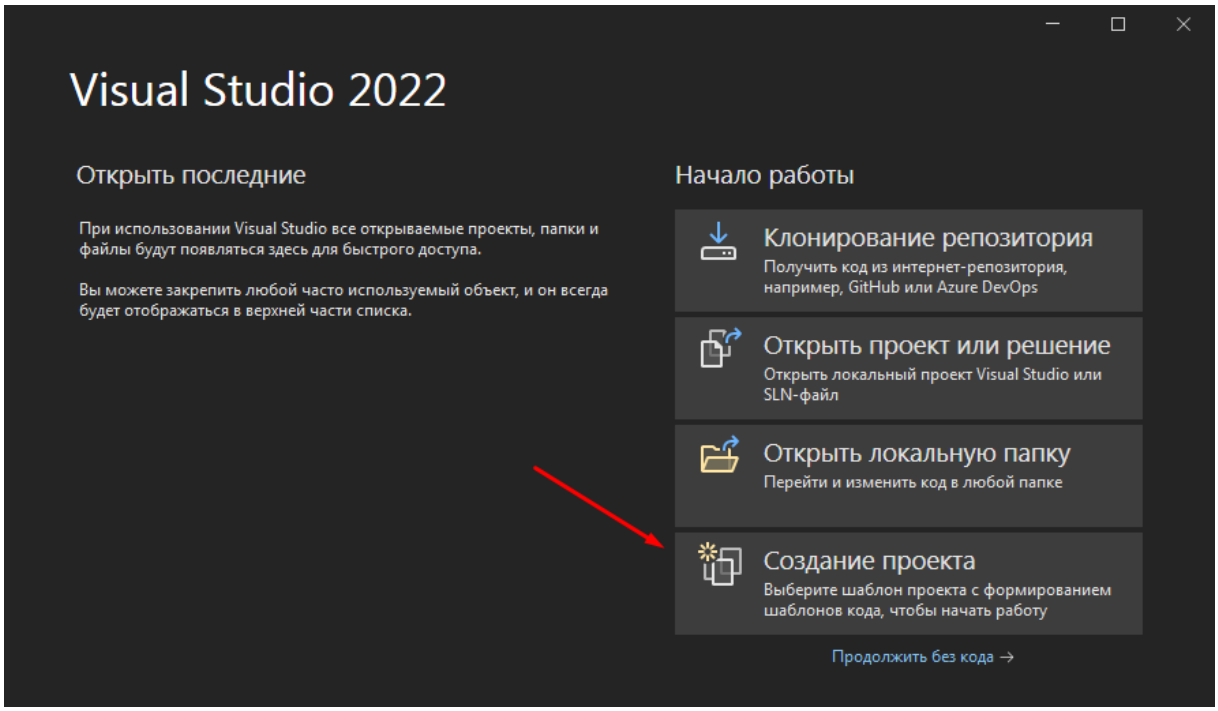

Рисунок 4.1 – Вікно створення нового проекту.

На сторінці налаштування (рис. 4.2) нового проекту Visual Studio [10, 11, 12] виконайте такі дії:

- введіть назву в поле "Ім'я проекту";
- вкажіть місце розташування проекту;
- Натисніть кнопку "Створити". (рис. 4.2)

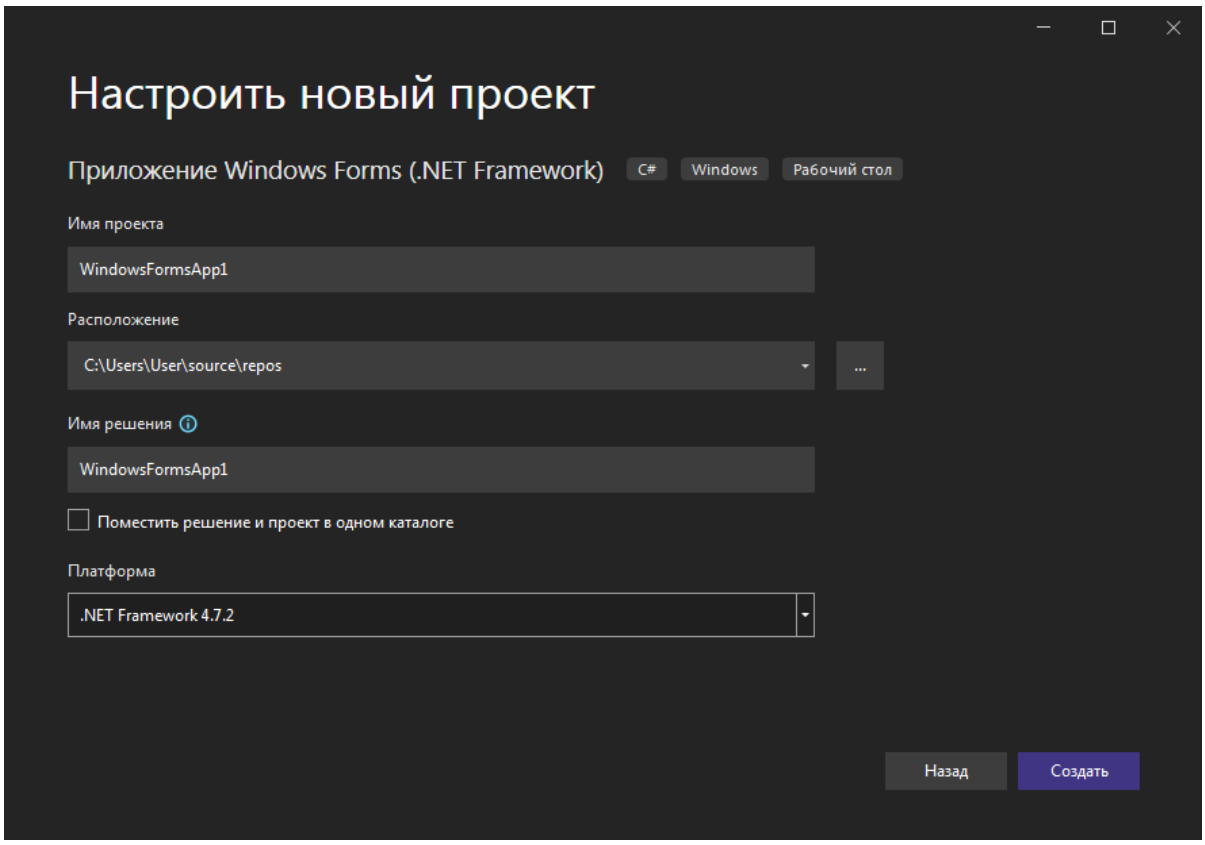

Рисунок 4.2 – налаштування нового проекту

Після створення проекту можемо додати такі елементи (рис. 4.3) на форму:

- Label (текст опису чи інформація під час виконання);
- Button (кнопка, при натисканні виникає подія);
- RadioButton (вибір певного параметра із групи);
- PictureBox (виведення зображення).

Для кожного елемента пишемо відповідний код та функції на мові програмування C# [13, 14, 15], перевіряємо на працездатність та робимо коригування. (рис. 4.4)

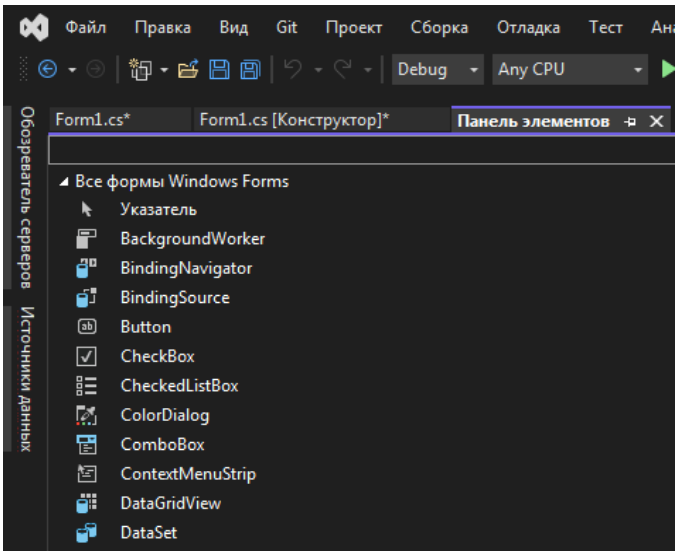

Рисунок 4.3 – панель елементів форми

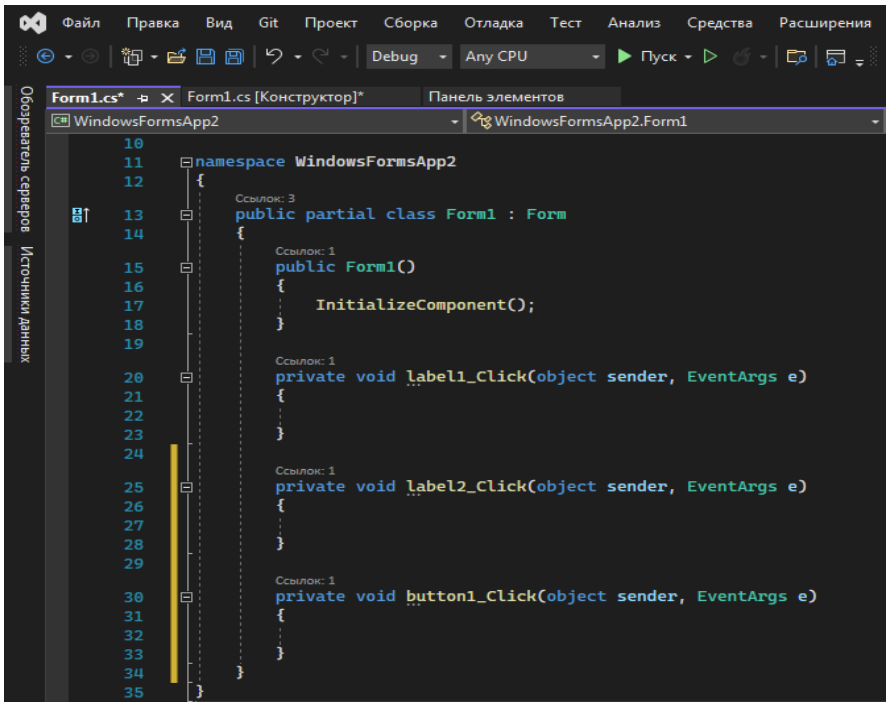

Рисунок 4.4 – вікно для написання коду елементам форми

4.3. Інструкція по використанню навчального тренажеру

Після запуску системи практичного навчання користувач переходить до головного вікна з інформацією, що містить:

- тему;
- інформацію про автора;
- інформацію про керівника;
- перехід до виконання. (рис. 4.5)

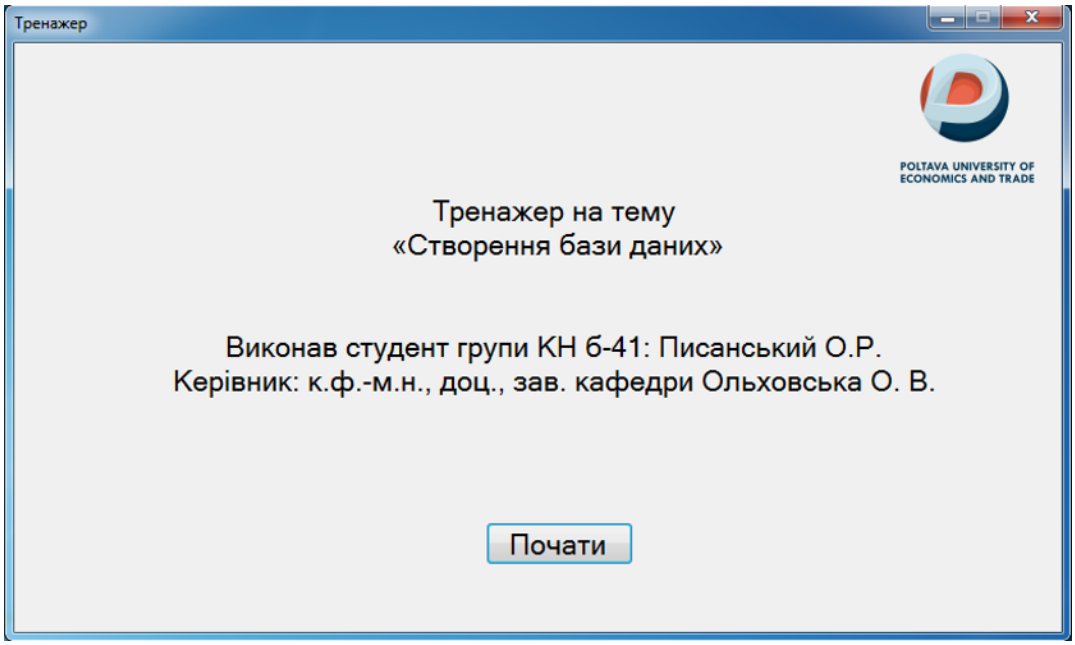

Рисунок 4.5 – головне вікно.

Після натиснення на кнопку «Почати» відбувається перехід до практичної частини тренажеру.

На кожному кроці виводиться завдання, необхідно вибрати відповідь. (рис. 4.6)

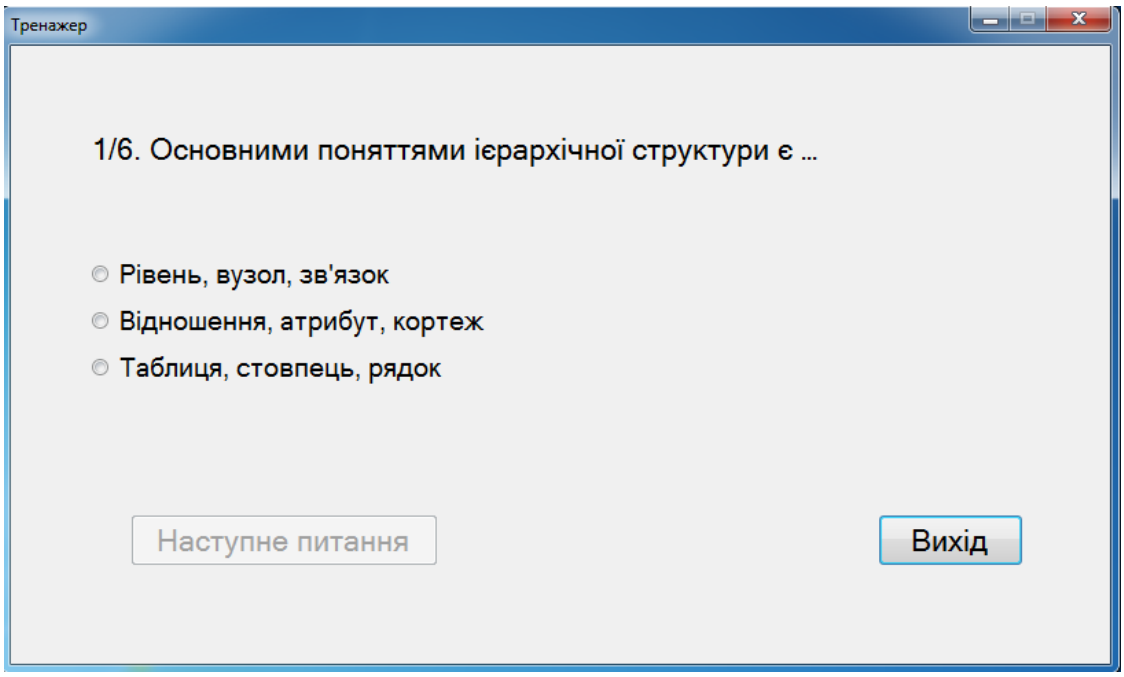

Рисунок 4.6 – зовнішній вигляд практичного завдання

Після вибору відповіді кнопка для переходу на наступне завдання розблокується. (рис. 4.7)

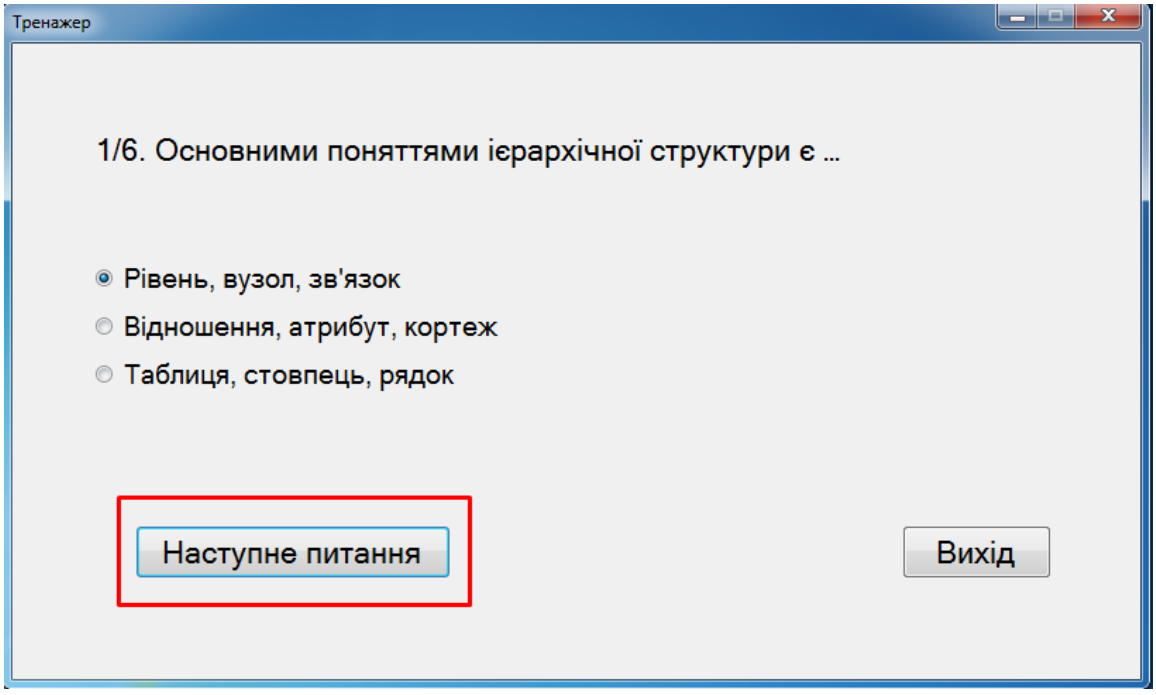

Рисунок 4.7 – практичне завдання після вибору відповіді

ВИСНОВКИ

Отже, одним з важливих етапів навчального процесу є доступ до навчального матеріалу, як правило матеріал має бути максимально легким до сприйняття і оптимально наповнений смисловим значенням і не переобтяжений зайвою інформацією.

Створення так званих віртуальних тренажерів – важливий етап при вирішенні проблеми організації навчання, де необхідні практичні навички. Основна перевага застосування віртуальних тренажерів полягає в тому, що вони можуть використовуватися як в навчальному процесі (при проведенні лабораторних робіт або для здійснення теоретичного допуску до них), так і для самостійного навчання студента. Застосування розробленої методики віртуальних практичних інтерактивних засобів навчальних дисциплін для дистанційного навчання дає змогу вирішити проблему впровадження інформаційних дистанційних технологій у навчальний процес з багатьох напрямів і спеціальностей підготовки.

Основні результати роботи:

- Проведено детальне вивчення теми на тему "Створення бази даних".
- Розроблено алгоритм для компонентів, які будуть використовуватися в програмному забезпеченні системи практичного навчання.
- Побудовано логічну, фізичну, блок-схему алгоритму.
- Створено програмне забезпечення, систему практичного навчання в середовищі Visual Studio з використанням мови програмування C#.
- Описано результати розробки системи практичного навчання.

В результаті виконання кваліфікаційної роботи було створено систему практичного навчання з теми «Створення бази даних» дисципліни «Розподілені інформаційно-аналітичні системи».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черненко О. О. Курсовий проєкт із фаху: методичні рекомендації щодо оформлення пояснювальних записок до курсового проєкту для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра, магістра / О. О. Черненко. – Полтава : ПУЕТ, 2022. – 58 с. – 1 електрон. опт. диск (CVD-ROM).
2. Організація дистанційної форми освіти [Електронний ресурс]. – Режим доступу: https://www.helsinki.org.ua/articles/orhanizatsiia-dystantsiynoi-formy-osvity-v-zzso-v-umovakh-voiennoho-stanu-na-shcho-potribno-zvernuty-uvahu/
3. Що таке база даних [Електронний ресурс]. – Режим доступу: https://apeps.kpi.ua/shco-take-basa-danykh
4. Типи баз даних: особливості, відмінності та приклади [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/types-of-databases/
5. Основні відомості про бази даних [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/uk-ua/topic/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%
6. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
7. Microsoft Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
8. C# development with Visual Studio [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022
9. C# Get Started [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/cs/cs_getstarted.php

10. Visual Studio and C# [Електронний ресурс]. – Режим доступу: https://www.halvorsen.blog/documents/programming/csharp/csharp.php
11. Створення програми на C# в Visual Studio 2015 [Електронний ресурс]. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9976
12. Getting Started With C# on Visual Studio Code [Електронний ресурс]. – Режим доступу: https://medium.com/@adebiyiadedotun9/getting-started-with-c-on-visual-studio-code-5a76dcca1110
13. Randolph Visual Studio 2010 for professionals // Randolph, Nick, Gardner, David, Minutillo, Michael, Anderson, Chris.: Trans. with English - M.: LLC "I.D. Williams", 2011. - 1184 p.
14. ReSharper: The Visual Studio Extension for .NET [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/resharper/
15. Install Visual Studio [Електронний ресурс]. – Режим доступу: https://www.csharptutorial.net/csharp-tutorial/install-visual-studio/

ДОДАТОК А. КОД ПРОГРАМИ

```
namespace WindowsFormsApplication1
{
   partial class Form2
   {
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

         this.button1 = new System.Windows.Forms.Button();
         this.checkBox1 = new System.Windows.Forms.CheckBox();
         this.checkBox2 = new System.Windows.Forms.CheckBox();
         this.checkBox3 = new System.Windows.Forms.CheckBox();
         this.label1 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // button1
         //
         this.button1.BackColor = System.Drawing.Color.White;
         this.button1.Cursor = System.Windows.Forms.Cursors.Hand;
         this.button1.Font = new System.Drawing.Font("Segoe UI", 9F,
System.Drawing.FontStyle.Bold);
         this.button1.ForeColor = System.Drawing.Color.Black;
         this.button1.Location = new System.Drawing.Point(599, 463);
         this.button1.Name = "button1";
         this.button1.Size = new System.Drawing.Size(148, 43);
         this.button1.TabIndex = 0;
         this.button1.Text = "Наступне питання";
         this.button1.UseVisualStyleBackColor = false;
         this.button1.Click += new System.EventHandler(this.button1_Click);
```

```
            //
            // checkBox1
            //
            this.checkBox1.AutoSize = true;
            this.checkBox1.Font = new System.Drawing.Font("Segoe UI Semibold", 13.8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.checkBox1.Location = new System.Drawing.Point(36, 132);
            this.checkBox1.Name = "checkBox1";
            this.checkBox1.Size = new System.Drawing.Size(663, 68);
            this.checkBox1.TabIndex = 1;
            this.checkBox1.UseVisualStyleBackColor = true;
            this.checkBox1.CheckedChanged += new
System.EventHandler(this.checkBox1_CheckedChanged);
            //
            // checkBox2
            //
            this.checkBox2.AutoSize = true;
            this.checkBox2.Font = new System.Drawing.Font("Segoe UI Semibold", 13.8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.checkBox2.Location = new System.Drawing.Point(36, 206);
            this.checkBox2.Name = "checkBox2";
            this.checkBox2.Size = new System.Drawing.Size(703, 68);
            this.checkBox2.TabIndex = 2;
            this.checkBox2.UseVisualStyleBackColor = true;
            this.checkBox2.CheckedChanged += new
System.EventHandler(this.checkBox2_CheckedChanged);
            //
            // checkBox3
            //
            this.checkBox3.AutoSize = true;
            this.checkBox3.Font = new System.Drawing.Font("Segoe UI Semibold", 13.8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.checkBox3.Location = new System.Drawing.Point(36, 280);
            this.checkBox3.Name = "checkBox3";
            this.checkBox3.Size = new System.Drawing.Size(617, 100);
```

```
            this.checkBox3.TabIndex = 3;

            this.checkBox3.UseVisualStyleBackColor = true;
            this.checkBox3.CheckedChanged += new
System.EventHandler(this.checkBox3_CheckedChanged);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Segoe UI", 14F,
System.Drawing.FontStyle.Bold);
            this.label1.Location = new System.Drawing.Point(224, 57);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(322, 32);
            this.label1.TabIndex = 4;

            //
            // Form2
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.White;
            this.ClientSize = new System.Drawing.Size(759, 518);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.checkBox3);
            this.Controls.Add(this.checkBox2);
            this.Controls.Add(this.checkBox1);
            this.Controls.Add(this.button1);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Name = "Form2";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Load += new System.EventHandler(this.Form2_Load_1);
            this.ResumeLayout(false);
            this.PerformLayout();
```

```
        }

        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.CheckBox checkBox2;
        private System.Windows.Forms.CheckBox checkBox3;
        private System.Windows.Forms.Label label1;
    }
}
```