

Міністерство освіти і науки України  
Державний заклад  
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Селіверстов Владислав Дмитрович

**РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ  
НАВЧАЛЬНИХ ЗАКЛАДІВ ІЗ ЗАСТОСУВАННЯМ КЛІЄНТ-  
СЕРВЕРНИХ СУБД**

**кваліфікаційна робота**

**здобувача вищої освіти першого (бакалаврського) рівня  
освітньої програми «Інженерія програмного забезпечення»  
за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис \_\_\_\_\_ Владислав СЕЛІВЕРСТОВ

Науковий керівник \_\_\_\_\_ Ольга СМАГІНА,  
кандидат педагогічних наук, доцент  
кафедри інформаційних технологій  
та систем

Завідувач кафедри \_\_\_\_\_ Микола СЕМЕНОВ,  
кандидат педагогічних наук, доцент  
кафедри інформаційних технологій  
та систем

Полтава – 2024

Міністерство освіти і науки України  
Державний заклад „Луганський національний університет  
імені Тараса Шевченка”

Інститут	Навчально-науковий інститут математики та інформаційних технологій
Кафедра, циклова комісія	Інформаційних технологій та систем
Рівень освіти	перший (бакалаврський)
Напрямок підготовки (спеціальність)	121 «Інженерія програмного забезпечення» (код, назва)

ЗАТВЕРДЖУЮ  
Завідувач кафедри ІТС  
М.А. Семенов

(підпис) (ініціали, прізвище)  
“ ” 2023 р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Селіверстову Владиславу Дмитровичу  
(прізвище, ім'я, по батькові )

**1. Тема проекту (роботи) Розробка інформаційно-довідкової системи для навчальних закладів із застосуванням клієнт-серверних СУБД**

Керівник кваліфікаційної роботи Смагіна О.О.  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету Від“ ” 2023 року №\_

**2. Строк подання студентом проекту (роботи)**

**3. Вихідні дані до роботи (проекту)**

Розроблено інформаційно-довідкову систему навчального закладу: на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ**

ІНФОРМАЦІЙНО- ДОВІДКОВИХ СИСТЕМ. ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ БАЗИ ДАНИХ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ.

РОЗРОБКА ДОДАТКУ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ  
ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

**5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)**

**6. Консультанти розділів проекту (роботи)**

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „\_\_\_\_\_” \_\_\_\_\_ 2023 р.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада )	
	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
	Попередній захист роботи на кафедрі	квітень	
	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

\_\_\_\_\_

підпис

В. Д. Селіверстов

(ініціали, прізвище)

Керівник проекту (роботи)

\_\_\_\_\_

підпис

О.О. Смагіна

## АНОТАЦІЯ

**Селіверстов В. Д.**

**Тема:** Розробка інформаційно-довідкової системи для навчальних закладів із застосуванням клієнт-серверних СУБД.

**Спеціальність:** 121 "Інженерія програмного забезпечення"

**Установа:** ЛНУ імені Тараса Шевченка, 2024 р.

**Бакалаврська робота містить:** 61 с., 18 рис., 25 табл., 3 додат., 25 джерел.

**Об'єктом дослідження** є технології розробки інформаційно- довідкових систем для навчальних закладів із застосуванням клієнт- серверних СУБД.

**Предметом дослідження** є інформаційно-довідкова система для навчальних закладів із застосуванням клієнт-серверних СУБД.

**Мета роботи** – аналіз технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД.

**Результати роботи.** У роботі досліджено теоретичні основи інформаційно-довідкових систем: розглянуто поняття ІДС, класифікація, структуру та складові підсистеми. Проаналізовано засоби проєктування та етапи створення інформаційно-довідкових систем. Визначені завдання, вимоги та обґрунтування розробки та розроблено технічне завдання на розробку ІДС. Розглянуто теоретичні підходи до технології клієнт-серверної архітектури, проаналізовано сучасні клієнт-серверні СУБД, які вільно розповсюджуються. Для розробки ІДС обрано FireBird. Під час методологічного проєктування розроблено інфологічну та даталогічну модель бази даних системи та розроблено фізичну модель бази даних school.gdb. Під час розробки клієнтського додатку ІДС були обґрунтовано застосовані технологія ADO.NET, мова програмування C #, середовище розробки Visual Studio. Розроблено інформаційно-довідкову систему навчального закладу: на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням.

**Ключові слова.** ІНФОРМАЦІО-ДОВІДКОВА СИСТЕМА; КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, БАЗА ДАНИХ, ТЕХНОЛОГІЯ, ЕТАПИ РОЗРОБКИ.

## ABSTRACT

**Seliverstov Vladyslav**

**Theme: Development of an information and reference system for educational institutions using client-server DBMS.**

**Speciality:** 121 "Software Engineering"

**Institution:** Luhansk Taras Shevchenko National University (LTSNU), 2024.

**Diploma work contains:** 61 pages, 18 Fig., 25 Table, 3 adj., 25 source.

**A research object is** technologies for developing information and reference systems for educational institutions using client-server DBMS.

**The article of research is** information and reference system for educational institutions using client-server DBMS.

**An aim of work is** analysis of technology development of information and reference systems for educational institutions using client-server DBMS.

**Job performances.** The paper examines the theoretical foundations of information and reference systems: the concept of IDS, classification, structure and components of subsystems are considered. Design tools and stages of creation of information and reference systems are analyzed. Tasks, requirements and justification of the development have been determined, and the technical task for the development of the IDS has been developed. Theoretical approaches to client-server architecture technology are considered, modern client-server DBMSs that are freely distributed are analyzed. FireBird was chosen for IDS development. During methodological design, an infological and datalogical model of the system database was developed, and a physical model of the school.gdb database was developed. During the development of the IDS client application, ADO.NET technology, the C # programming language, and the Visual Studio development environment were reasonably applied. The information and reference system of the educational institution has been developed: based on the client-server architecture, which fully meets the requirements and tasks.

**Keywords.** INFORMATION AND REFERENCE SYSTEM; CLIENT - SERVER ARCHITECTURE, DATABASE, TECHNOLOGY, DEVELOPMENT STAGES.

**ІТС. ІПЗ4.0524-ВІП**  
**ВІДОМІСТЬ ПРОЄКТУ. РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ**  
**СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ ІЗ ЗАСТОСУВАННЯМ**  
**КЛІЄНТ-СЕРВЕРНИХ СУБД.**

[illegible]

Міністерство освіти і науки України  
Державний заклад «Луганський національний університет  
імені Тараса Шевченка»  
Факультет (інститут) Навчально-науковий інститут математики та  
інформаційних технологій  
(повна назва)  
Кафедра Інформаційних технологій та систем  
(повна назва)

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на виконання програмної розробки (ПР) :  
**"РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ  
НАВЧАЛЬНИХ ЗАКЛАДІВ ІЗ ЗАСТОСУВАННЯМ КЛІЄНТ-  
СЕРВЕРНИХ СУБД"**  
**ІТС. ІПЗ4.0524-02-ТЗ**

**ПОГОДЖЕНО**  
Керівник кваліфікаційної роботи

\_\_\_\_ Смагіна О.О. \_\_\_\_\_

“ \_\_\_\_ ” \_\_\_\_\_ 2024р.

**ВИКОНАВЕЦЬ**  
Студент групи 4 ІПЗ

\_\_\_\_ Селіверстов В. Д. \_\_\_\_\_

“ \_\_\_\_ ” \_\_\_\_\_ 2024р.

Полтава 2024

## ЗМІСТ

ВСТУП .....	3
1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ .....	3
2. ПРИЗНАЧЕННЯ ПРОДУКЦІЇ .....	3
3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ .....	4
4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ .....	5
5. ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ .....	5
6. ЕТАПИ ВИКОНАННЯ ПР. ....	6
7. ПРИЙМАННЯ.....	6
8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО.....	7

## **ВСТУП**

**1.1 Найменування:** Інформаційно-довідкова система для навчальних закладів із застосуванням клієнт-серверних СУБД.

**1.2 Шифр ПР:** АДР-23-ПІ.5

**1.3 Підстава до виконання ПР:** Підставою для виконання даної розробки є необхідність розробки інформаційно-довідкової системи для навчальних закладів із застосуванням клієнт-серверних СУБД.

**1.4 Терміни розробки:**

1.4.1 Початок 30 жовтня 2023р.

1.4.2 Закінчення 30 березня 2024р.

**1.5 Фінансується** за рахунок коштів замовника. Умови фінансування – згідно договору №16/а та протоколу погодження ціни № 16/б.

## **1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ**

1.1. Розробляємий програмний комплекс повинен автоматизувати процеси формування довідкової інформації в навчальному закладі. **До складу об'єкту**, що створюється повинно входити:

1.1.1. Програмний комплекс, що розробляється

1.2. **До вхідної інформації** належать вимоги замовника щодо певної інформації.

1.3. **До вихідної інформації** належать звіти про склад учнів класу, навантаження викладачів.

## **2. ПРИЗНАЧЕННЯ ПРОДУКЦІЇ**

**2.1. Призначення:** Інформаційно-довідкова система (ІДС) призначена для формування довідкової інформації за допомогою інформаційних технологій в загальноосвітньому закладі.

**2.2. Основні критерії ефективності**

2.2.1. Програмний комплекс повинен:

2.2.1.1. надання інформації про учнів, їх батьків, предмети та факультативи, що вивчають учні, результати навчальної діяльності кожного учня, викладацький персонал, навантаження вчителів;

- 2.2.1.2. додавання, видалення, оновлення даних про учнів, батьків учнів, вчителів, навчальний процес, відвідування занять учнями тощо;
- 2.2.1.3. ведення довідників, які використовуватимуться базою даних, а саме: національність, відомості про батьків, стан здоров'я учнів, кваліфікація вчителів та навчальний заклад, який вони закінчили.
- 2.2.1.4. формування даних про учня, його успішність, інформація про відвідування занять за ознакою «ПБ», пошук даних про учнів за ознакою «дата народження»;
- 2.2.1.5. формування даних про вчителя, його навчальне навантаження за ознакою ПБ, пошук даних про викладачів за ознакою «категорія»;
- 2.2.1.6. формування звітів про склад учнів класу, навантаження викладачів.

### **2.3. Основні функції оператора**

- 2.3.1. Запустити програмний комплекс;
- 2.3.2. Заповнювати базу даних комплексу;
- 2.4. Основною функцією програмного комплексу є автоматизація послідовності дій оператора.

## **3. ОСНОВНІ ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ**

### **3.1. Загальні вимоги**

- 3.1.1. програмний комплекс працює під операційною системою WINDOWS 10/7;
- 3.1.2. вимоги до апаратного забезпечення персонального комп'ютеру – не передбачаються та можуть встановлюватися розробником програмного комплексу;
- 3.1.3. програмний комплекс повинен мати зручний інтерфейс;
- 3.1.4. Інформаційно-довідкова система повинна забезпечувати формування, збереження, редагування основних масивів довідкової інформації;
- 3.1.5. ІДС повинна забезпечувати створення інформаційного документу як у електронному, так й у друкованому (паперовому) вигляді.
- 3.1.6. ІДС повинна функціонувати в мережевому просторі та працювати в багатокористувацькому режимі.

### **3.2. Додаткові вимоги**

3.2.1. мова програмування C#, СУБД Firebird.

3.2.2. вимоги до ліцензійного ПЗ не передбачаються та вирішуються замовником

### **3.3. Вимоги до складу та архітектури**

3.3.1. розробник самостійно обирає склад та виконує розробку архітектури ПР

3.3.2. особливих умов до складу та архітектури ПР не передбачено

### **3.4. Вимоги до якості та надійності**

3.4.1. програмний комплекс повинен надійно працювати

3.4.2. розробник обирає технічні характеристики персонального комп'ютера, налаштовує системне програмне забезпечення.

3.4.3. розробник гарантує роботу програмного комплексу без збоїв та пере налаштувань.

3.4.4. виконавець гарантує придбання додаткового обладнання (кондиціонер, UPS) за власні кошти.

### **3.5. Вимоги до експлуатації**

3.5.1. розробник використовує персональний комп'ютер, на якому програмний комплекс повинен надійно працювати.

3.5.2. Персональний комп'ютер буде задіяно у розрахунках та буде встановлено у приміщенні обчислювального центру.

## **4. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ**

Вартість робіт по розробці даної ПР визначається згідно договору на розробку. Вартість запропонованих аналогів повинна забезпечити економічну доцільність їх застосування.

## **5. ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ**

В процесі розробки програмного комплексу можливе використання мови програмування C # та технології доступу до даних ADO.NET, виберемо середовище Visual Studio.

5.1. Вимоги до екологічної безпечності під час експлуатації.

Не пред'являються.

5.2. Спеціальні вимоги до кінцевого продукту.

Не пред'являються.

5.3. Вимоги до безпеки для населення під час експлуатації продукції.

Не пред'являються.

## **6. ЕТАПИ ВИКОНАННЯ ПР.**

Етапи виконання ПР можуть уточнюватись згідно календарного плану робіт по узгодженню між замовником та виконавцем

№	Етапи виконання роботи	Термін виконання та обсяг робіт	звітні матеріали
1	Аналіз розробки програмного комплексу та розробка першої версії. Аналіз вимог. Розробка структури. Попереднє тестування		Частковий програмний комплекс на ЕОМ замовника, що виконує всі основні функції та звітна документація п.8.2
2	Коректування структури. Розробка допоміжних функцій. Розробка остаточної версії програмного комплексу та його опрацювання. Тестування		Готовий програмний комплекс на ЕОМ замовника та звітна документація п.8.2
3	Доопрацювання окремих модулів та навчання користувачів. Розробка звітних матеріалів згідно п.8 цього ТЗ		звітні матеріали згідно пункту 8

## **7. ПРИЙМАННЯ**

7.1. Необхідні вимоги для впровадження ПР та завершення робіт.

Оцінка результатів розробки і доцільність її продовження здійснюється замовником по представленню наступних матеріалів:

- встановлений програмний комплекс на ЕОМ замовника;
- перелік файлів на резервному носії;
- стислий опис роботи ПР та опис всіх файлів, які необхідні для роботи ПР.

- перелік документів
  - Технічне завдання
  - Пояснювальна записка
  - Програма та методика тестування
  - Керівництво користувача

7.2. Перелік звітних документів, необхідних для прийняття етапів роботи:

- стислий опис результатів етапу у вигляді анотованого звіту(для 1 та 2 етапів);
- частковий програмний комплекс на ЕОМ замовника згідно календарного плану робіт;
- акт приймання продукції.

Звітні матеріали подаються у вигляді звітів на папері по ГОСТ 7.32-91

7.3. Загальний перелік до приймання звітних документів, макетів, експериментальних зразків.

До приймання пред'являються: акт здачі-приймання продукції, акт впровадження ПР.

7.4.Тестування ПР

Тестування виконується до "Програми та методики тестування", яка розробляється виконавцем та затверджується замовником

## **8. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО.**

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням доповнень до ТЗ.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**  
**ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

Навчально-науковий інститут математики та інформаційних  
технологій

(назва факультету, інституту)

Інформаційних технологій та систем

(назва кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

БАКАЛАВРА

(освітньо-кваліфікаційний рівень)

на тему: **РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ**  
**НАВЧАЛЬНИХ ЗАКЛАДІВ ІЗ ЗАСТОСУВАННЯМ КЛІЄНТ-**  
**СЕРВЕРНИХ СУБД**

Виконав: студент 4 курсу  
напряму підготовки (спеціальності)  
121 «Інженерія програмного  
забезпечення»

(шифр і назва напряму підготовки, спеціальності)

Селіверстов В. Д.

(прізвище та ініціали)

Керівник Смагіна О.О.

(прізвище та ініціали)

Рецензент Козуб Ю. Г.

(прізвище та ініціали)

Полтава – 2024 року

## **ЗМІСТ**

<b>ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ .....</b>	<b>3</b>
<b>ВСТУП.....</b>	<b>4</b>
<b>РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ ІНФОРМАЦІЙНО-ДОВІДКОВИХ СИСТЕМ .....</b>	<b>6</b>
1.1. Теоретичні основи інформаційно-довідкових систем .....	6
1.2. Технологія проєктування та розробки інформаційно-довідкових систем..	10
1.3. Передпроектний етап розробки інформаційно-довідкової системи для навчального закладу .....	12
Висновки до розділу .....	15
<b>РОЗДІЛ 2. ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ БАЗИ ДАНИХ ІНФОРМАЦІНО-ДОВІДКОВОЇ СИСТЕМИ.....</b>	<b>17</b>
2.1. Клієнт – серверна архітектура СУБД .....	17
2.2. Порівняльний аналіз СУБД клієнт – серверної архітектури.....	21
2.3. Проєктування моделі бази даних інформаційно-довідкової системи .....	24
Висновки до розділу .....	34
<b>РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ .....</b>	<b>36</b>
3.1. Аналіз засобів та технологій розробки додатку .....	36
3.2. Розробка та реалізація програмного додатку.....	42
3.3. Тестування додатка.....	54
Висновки до розділу .....	57
<b>ВИСНОВКИ .....</b>	<b>58</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>	<b>59</b>
<b>ДОДАТОК А.....</b>	<b>62</b>
<b>ДОДАТОК Б .....</b>	<b>63</b>
<b>ДОДАТОК В .....</b>	<b>76</b>

## **ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

АІДС – автоматизована інформаційно-довідкова система

АІС – автоматизована інформаційна система

БД – база даних

ІДС – інформаційно-довідкова система

СУБД – система управління базами даних

## ВСТУП

**Актуальність теми.** У сучасних умовах, які характеризуються активізацією інформаційних процесів у всіх сферах суспільного життя, зокрема в освіті, набуває актуальності проблема інформаційного забезпечення освітнього закладу, від якого залежить ефективна організація навчального процесу.

На відміну від роботи з довідковою інформацією на паперових носіях перехід до автоматизованих інформаційно-довідкових систем забезпечує ряд переваг. Електронні документи можуть одночасно використовуватися співробітниками в рамках однієї робочої групи, відділу або всього закладу. Доступ до них здійснюється дуже швидко. Прискорений доступ до стратегічної інформації разом із значною економією коштів може забезпечити і важливі організаційні переваги.

Впровадження автоматизованих інформаційно-довідкових систем дозволяє вирішити завдання, що найбільш часто зустрічаються: забезпечення більш ефективного управління за рахунок автоматичного контролю виконання, прозорості діяльності організації на всіх рівнях; підтримку ефективного накопичення, керування і доступу до інформації; забезпечення кадрової гнучкості за рахунок більшої формалізації діяльності кожного співробітника і можливості збереження всієї передісторії його діяльності; виключення паперових документів з внутрішнього обороту підприємства і пов'язану з цим економію ресурсів за рахунок скорочення витрат на управління потоками електронної інформації в організації; істотне спрощення і здешевлення збереження паперових документів за рахунок наявності оперативного електронного архіву. Зазначене обумовлює актуальність обраної теми.

**Метою роботи** є аналіз технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД.

**Об'єктом дослідження** є технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД.

**Предметом дослідження** є інформаційно-довідкова система для навчальних закладів із застосуванням клієнт-серверних СУБД.

Відповідно до предмета, мети було визначено основні **завдання дослідження:**

- дослідити інформаційно-довідкову систему для навчальних закладів;
- проаналізувати етапи розробки інформаційно-довідкових систем;
- визначити завдання та вимоги до інформаційно-довідкової системи;
- розглянути можливості технології клієнт-серверної архітектури;
- проаналізувати існуючі СУБД клієнт-серверної архітектури та обґрунтувати вибір СУБД для подальшої розробки бази даних;
- змодельовати базу даних інформаційно-довідкової системи;
- розробити клієнтський додаток інформаційно-довідкової системи для навчальних закладів із застосуванням клієнт-серверних СУБД.

Для вирішення завдань дослідження використано такі **методи дослідження:** *теоретичні:* аналіз наукової літератури, узагальнення та систематизація теоретичних положень про інформаційно-довідкові системи для навчальних закладів, моделювання інформаційних процесів; *емпіричні:* порівняльний аналіз можливостей інструментів розробки програмних додатків; *експериментальні:* тестування розробленої системи.

До складу роботи входять три розділи, які висвітлюють питання технології розробки інформаційно-довідкових систем для навчальних закладів, підходи до проєктування баз даних клієнт-серверної архітектури.

Практичним результатом роботи є розроблена інформаційно-довідкова система навчального закладу: "Дніпровська загальноосвітня школа І - ІІІ ступенів №5" на базі клієнт- серверної архітектури.

# РОЗДІЛ 1.

## АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ ІНФОРМАЦІЙНО- ДОВІДКОВИХ СИСТЕМ

### 1.1. Теоретичні основи інформаційно-довідкових систем

Зростання обсягів інформації, прискорення і ускладнення її обліку викликають необхідність автоматизації процесу зберігання, обробки та відображення інформації.

Комплекс засобів, що дозволяє забезпечити безперервне протікання інформаційних процесів становлять автоматизовану інформаційну систему (АІС) - це [1, с. 9].

Різновидом АІС є автоматизована інформаційно-довідкова система. (АІДС). Інформаційно-довідкові системи орієнтовані більшою мірою на збір, зберігання і видачу за запитом користувача формалізованої інформації економічного, технічного або технологічного характеру. Можна вважати, що ІДС орієнтовані на роботу з конкретизованими даними цифрового або текстового типу.

Залежно від предметної галузі АІДС можуть дуже сильно відрізнятися за своїми функціями та способами реалізації. Однак можна виділити ряд властивостей, які є загальними для всіх інформаційно-довідкових систем, а саме: вони призначені для збору, зберігання та обробки інформації. Тому в основі будь-якої з них лежить середовище зберігання та доступу до даних (база даних); вони орієнтуються на кінцевого користувача, які часто мають низький рівень інформаційної культури та володіють навичками роботи за комп'ютерами на початковому рівні.

Тому клієнтські програми інформаційної системи повинні володіти простим та зручним графічним інтерфейсом, який надає кінцевому користувачеві всі необхідні для роботи функції та запобігає виконання зайвих дій [11, с. 15].

Щодо визначення завдань, які висуваються до інформаційно- довідкових систем, слід розглянути основні завдання інформаційних систем:

1. Як і ІС, інформаційно-довідкові системи призначені для пошуку,

обробки та збереження інформації за допомогою комп'ютерної техніки. Використання комп'ютерів дозволяє забезпечити більш швидку та надійну обробку інформації, зекономити людський час на роботу з даними та уникнути властивих людині випадкових помилок.

2. Для збереження та обробки інформації ІДС повинні мати відповідні засоби для роботи з ними. У результаті розвитку більшості таких систем у них виділився окремий компонент, який являє собою різновид системи управління базами даних.
3. ІДС повинні мати певні процедури, за допомогою яких можна автоматизувати процес формування звітної документації.
4. Наявність можливостей виконання різноманітних інформаційних запитів користувачів до інформаційних сховищ та баз даних, а також підтримка спеціальних мов запитів для систем такого типу [10].

Слід зазначити, що завдання, які повинні вирішуватися конкретною інформаційно-довідковою системою, залежать від тієї прикладної галузі, для якої призначена ця система. Галузі застосування інформаційних додатків можуть бути доволі різноманітними: банківська справа, управління виробництвом, медицина, транспорт, освіта тощо.

Використання АІДС в системах організаційного управління забезпечує:

- інформаційно-довідкове обслуговування;
- автоматизацію діловодства;
- розвинений діалог між користувачем і ЕОМ;
- формування і ведення локальних баз даних і використання централізованої бази даних при наявності обчислювальної мережі;
- надання різних сервісних послуг користувачам на робочому місці.

В основу *класифікації АІДС* може бути покладено ряд класифікаційних ознак. Розрізняють автоматизовані інформаційно-довідкові системи з вигляду запиту і формою подання результату. Запит може бути стандартний і довільний. Результат може бути представлений або довідкою стандартної форми, або

форма проєктується в довільному вигляді за бажанням користувача в момент обробки його запиту.

Залежно від характеру роботи з інформацією розрізняють наступні види АІДС:

- автоматизовані архіви (АА);
- автоматизовані системи діловодства (АСД);
- геоінформаційні системи (ГІС);
- автоматизовані довідники (АД) і каталоги (АК) та ін.

За способом зберігання даних поділяють наступні АІДС:

- фактографічні (дані зберігаються в структурованому вигляді). У фактографічних АІС реєструються факти - конкретні значення даних (атрибутів) про об'єкти реального світу;
- документальні (в текстовому вигляді). Базу даних таких систем утворює сукупність неструктурованих текстових документів (статті, книги, реферати, тексти законів) і графічних об'єктів, забезпечена тим чи іншим формалізованим апаратом пошуку.

У складі АІДС можна виділити три основних технологічних процеси, представлені на рис. 1.1.



Рис. 1.1. Основні технологічні процеси АІДС

*Організаційно-технологічна підсистема збору і введення інформації* забезпечує відбір і накопичення даних в інформаційну систему і включає сукупність джерел інформації, організаційно-технологічні ланцюжки відбору інформації для накопичення в системі. Без правильно організованої, оперативно і ефективно діючої організаційно-технологічної підсистеми збору інформації

неможлива ефективна організація функціонування всієї інформаційної системи в цілому.

*Підсистема подання та обробки інформації* становить ядро інформаційно-довідкової системи і є відображенням уявлення розробниками і абонентами системи структури та картини предметної області, відомості про яку повинна відображати інформаційно-довідкова система. Підсистема подання та обробки інформації є одним з найбільш складних компонентів при розробці інформаційної системи.

Інформаційним ядром цієї підсистеми, або, інакше кажучи, внутрішнім носієм знань про предметну галузь є база даних (БД).

*Нормативно-функціональна підсистема виведення інформації* визначає користувачів, або інакше абонентів, системи, реалізує цільовий аспект призначення і виконання завдань інформаційної системи.

*Структуру автоматизованої інформаційно-довідкової системи* поділяють на дві складові: функціональну частину, яка відображатиме цілі і завдання системи, і частину, яка забезпечує виконання цих завдань та містить засоби вирішення завдань.

Частина, яка забезпечує АІДС, складається з інформаційного, програмного, технічного, організаційного забезпечення тощо.

До *інформаційного забезпечення* належать масиви інформації, що зберігається в базах даних на дискових накопичувачах. Сюди ж відноситься і СУБД.

До *технічного забезпечення* відноситься комплекс технічних засобів, які забезпечують вирішення поставлених завдань, а також засоби зв'язку з іншими АІС, які працюють в загальній мережі об'єкта, а також інші засоби зв'язку (телефон, факс та ін.).

*Програмне забезпечення* включає операційні системи, сервісні програми, стандартні програми користувачів і пакети прикладних програм, виконані за модульним принципом і орієнтовані на вирішення певного класу задач, обумовленого призначенням АІДС. У міру необхідності в програмне забезпечення включаються також пакети програм для роботи з графічною

інформацією.

*Організаційне забезпечення АІДС* має на меті організацію їх функціонування, розвитку, підготовки кадрів і адміністрування (планування роботи, облік, контроль, аналіз, регулювання, документальне оформлення прав і обов'язків користувачів АІДС).

Отже, інформаційно-довідкові системи є одним із видів інформаційних систем, які призначені для збереження, пошуку та обробки інформації. Створення АІДС забезпечує такі переваги розподіленої обробки даних, як більш низька вартість, висока надійність, поєднання автономного і розрахованого на багато користувачів режимів роботи і т. п.

## **1.2. Технологія проєктування та розробки інформаційно-довідкових систем**

*Проєктування АІДС* - процес створення і впровадження проєктів комплексного вирішення виробничих завдань за допомогою інформаційних технологій. Однією зі складових процесу проєктування є детальна розробка окремих проєктних рішень, їх аналіз, апробація і впровадження.

*Методи проєктування АІДС* включають: індивідуальне (оригінальне), типове проєктування, автоматизоване проєктування.

*Індивідуальне проєктування* характеризується тим, що всі види робіт для різних об'єктів виконуються за індивідуальними проєктами. У процесі індивідуального проєктування застосовуються свої оригінальні методики і засоби проведення робіт. Склад робіт на всіх етапах обстеження, проєктування і впровадження створюються для конкретного об'єкта. Для цього методу проєктування характерні висока трудомісткість, тривалі терміни проєктування.

*Типове проєктування* - розбиття системи на безліч складових компонентів і створення для кожного з них закінченого проєктного рішення, яке при впровадженні прив'язується до конкретних умов об'єкта. Залежно від декомпозиції розрізняють: елементне проєктування, підсистемне, об'єктне.

Під час *елементного методу проєктування* вся система розподіляється на кінцеве безліч елементів, кожен з яких є типовим. Елементами можуть бути проєктні рішення з інформаційного, технічного, програмного видів забезпечення.

*Підсистемний метод проєктування* характеризується більш високим ступенем інтеграції елементів ІДС.

Автоматизоване проєктування здійснюється на основі CASE-засобів. Виділяються кілька етапів створення АІДС:

**I етап - Передпроєктний** (обстеження, складання звіту, техніко-економічного обґрунтування і технічного завдання, формування вимог користувача до АІДС, розробка документації на АІДС).

Перш ніж починати проєктування, необхідно виконати обстеження об'єкта, для якого створюється ІС. Це досить важливий етап, так як дозволяє виділити характерні особливості об'єкта, які слід врахувати в характеристиках розробляється ІС і які визначають подальшу роботу з проєктування. Від якості передпроєктного обстеження багато в чому залежить, чи доведеться в подальшому переглядати основні концепції створюваної ІДС і вносити в неї принципові зміни, що завжди є трудомістким завданням.

Обстеження зводиться до аналізу існуючої системи і об'єкта автоматизації, для якого створюється система. На підставі вивчення об'єкта формується перелік завдань, які повинна вирішувати ІС. Результатом діяльності на цьому етапі буде сформоване технічне завдання, яке обґрунтування розробки та матеріали, що дають уявлення про склад і функціонування ІДС, і включає в себе:

- загальну характеристику об'єкта, для якого створюється ІДС;
- опис вимог до ІС;
- використовуваний комплекс технічних засобів;
- опис і постановку вирішення завдань, що входять в ІС;
- опис стандартного програмного забезпечення;
- опис організації інформаційної бази і т. д.

**II етап - Проєктний** (розробка проєктних рішень системи і її частин, розробка або адаптація програм).

Цей етап складається з двох підетапів:

1) **Етап технічного проєктування** - формуються проєктні рішення із забезпечувальної і функціональної частин інформаційної системи, здійснюється

постановка задачі і блок-схеми і їх рішення.

Саме на цьому етапі відбувається проектування інформаційних моделей баз даних системи – побудова інфологічної та даталогічної моделей БД, які відображають всю концепцію даних системи з їх якостями та асоціаціями.

2) ***Етап робочого проектування*** - здійснюється розробка фізичної бази даних, розробка програмного коду. Цей етап передбачає й доведення системи, коригування структури, створення різної документації: на поставку, на установку технічних засобів, інструкції по експлуатації, посадові інструкції тощо.

**III етап - Впровадження.** Цей етап передбачає підготовку до впровадження, проведення дослідних випробувань, тестувань інформаційно-довідкової системи, здачу в промислову експлуатацію, підготовку персоналу, проведення навчання персоналу.

**IV етап - Супровід і аналіз функціонування.** Цей етап складається з виявлення проблем, внесення змін в проєктні рішення і існуючі ІС.

Основними учасниками процесу створення автоматизованої інформаційно-довідкової системи є підприємство-замовник, для якого вона створюється і підприємство-розробник, що виконує роботи з проектування ІДС. Юридичні та організаційні взаємини конкретно замовників і розробників регулюються укладеними між ними договорами.

### **1.3. Передпроектний етап розробки інформаційно-довідкової системи для навчального закладу**

Відповідно до етапів розробки автоматизованої інформаційно-довідкової системи було проведено обстеження об'єкта автоматизації - інформаційно-довідкової системи навчального закладу: "Дніпровська загальноосвітня школа І - ІІІ ступенів №5"

На підставі вивчення об'єкта було сформовано цілі, завдання, вимоги до інформаційно-довідкової системи. Результатом цієї діяльності було розроблене технічне завдання.

**Технічне завдання на розробку інформаційно-довідкової системи для загальноосвітнього закладу** матиме наступні розділи.

## **1. Призначення, цілі та завдання:**

Інформаційно-довідкова система (ІДС) призначена для формування довідкової інформації за допомогою інформаційних технологій в загальноосвітньому закладі: "Лисичанська загальноосвітня школа І - ІІІ ступенів №5 Лисичанської міської ради Луганської області".

*Мета ІДС:* автоматизація процесів формування довідкової інформації в навчальному закладі: "Лисичанська загальноосвітня школа І - ІІІ ступенів №5 Лисичанської міської ради Луганської області".

Застосування *ІДС* сприятиме підвищенню ефективності організації навчального процесу та створенню єдиного інформаційного простору для введення, обробки, аналізу, збереження документів.

*Завдання, які вирішуються за допомогою ІДС :*

- надання інформації про учнів, їх батьків, предмети та факультативи, що вивчають учні, результати навчальної діяльності кожного учня, викладацький персонал, навантаження вчителів;
- додавання, видалення, оновлення даних про учнів, батьків учнів, вчителів, навчальний процес, відвідування занять учнями тощо;
- ведення довідників, які використовуватимуться базою даних, а саме: національність, відомості про батьків, стан здоров'я учнів, кваліфікація вчителів та навчальний заклад, який вони закінчили;
- формування даних про учня, його успішність, інформація про відвідування занять за ознакою «ПІБ», пошук даних про учнів за ознакою «дата народження»;
- формування даних про вчителя, його навчальне навантаження за ознакою ПІБ, пошук даних про викладачів за ознакою «категорія»;
- формування звітів про склад учнів класу, навантаження викладачів.

## **2. Вимоги до ІДС**

### *2.1. Загальні вимоги*

Інформаційно-довідкова система повинна забезпечувати формування, збереження, редагування основних масивів довідкової інформації.

ІДС повинна забезпечувати створення інформаційного документу як у

електронному, так й у друкованому (паперовому) вигляді.

ІДС повинна функціонувати в мережевому просторі та працювати в багатокористувацькому режимі.

## *2.2. Вимоги до можливостей налаштування та захисту ІДС*

З метою захисту інформації у інформаційно-довідкової системі необхідно передбачити відповідні заходи. Доступ до системи повинен здійснюватися за допомогою введення логіна та пароля, що дасть можливість захистити інформацію від несанкціонованого доступу.

Система повинна забезпечувати гнучкі можливості налаштування, а саме встановлення шляху доступу до бази даних, додавання нового користувача або видалення існуючого.

## *2.3. Вимоги до програмного та технічного забезпечення ІДС*

В основі реалізації можливостей ІДС повинна знаходитися технологія реляційних баз даних, що реалізує систематизоване сховище даних певної предметної галузі. Реляційні бази даних дозволять здійснювати контроль за надмірністю, несуперечливістю, цілісністю даних.

База даних повинна мати клієнт-серверну архітектуру, що забезпечить централізоване сховище даних на серверній частині та паралельну багатокористувацьку обробку даних.

Клієнт-серверна система складається з трьох основних компонент: програмне забезпечення сервера; програмне забезпечення кінцевого користувача (клієнта); проміжне програмне забезпечення.

Клієнтська частина ІДС повинна функціонувати під управлінням операційної системи Windows.

Комплекс технічних засобів обробки даних для вирішення прикладних завдань повинен включати такі основні компоненти:

- серверні платформи в складі ЛОМ, які підтримують функціонування СУБД;
- робочі станції, які забезпечують роботу користувачів на відповідних робочих місцях;
- засоби, що забезпечують гарантоване безперебійне живлення;

- телекомунікаційне обладнання, яке підтримує інформаційний обмін між користувачами.

Засоби безперебійного живлення повинні забезпечувати стабільне електроживлення в межах технічних характеристик обчислювальних засобів і периферійного обладнання. Електроживлення всіх складових частин сайту повинно виконуватися від електричної мережі змінного струму 220 В частотою 50 Гц.

### **3. Склад і зміст робіт**

Створення ІДС відбувається за наступною черговістю робіт:

- аналіз предметної галузі;
- вибір СУБД та програмних засобів проєктування БД;
- розробка інфологічної, даталогічної, фізичної моделей БД;
- розробка клієнтського додатку ІДС, що функціонує з серверною БД;
- тестування ІДС.

### **4. Порядок контролю і приймання ІДС**

Для ІДС встановлюються такі види контролю:

- попередній контроль (1 етап);
- підсумковий контроль (2 етап).

Попередній контроль повинен проводитися з метою визначення працездатності ІДС, кількісних та якісних характеристик ІДС, коригування подальших етапів розробки.

Приймання ІДС (підсумковий контроль) повинна проводитися комісією на підставі результатів попереднього контролю. Склад комісії, загальні вимоги з приймання робіт, порядок узгодження і затвердження приймальної документації повинні визначатися Замовником та узгоджуватися з виконавцем.

### **Висновки до розділу**

В першому розділі дипломної роботи було розглянуто аналіз технології розробки інформаційно-довідкових систем для навчального закладу, а саме:

- досліджено теоретичні основи інформаційно-довідкових систем: розглянуто поняття ІДС, класифікація, структуру та складові

підсистеми. Встановлено, що застосування ІДС дозволить ефективно здійснювати формування довідкової інформації й організацію роботи з нею;

- проаналізовано засоби проєктування та етапи створення інформаційно-довідкових систем. Встановлено, що технологія розробки ІДС складається з наступних етапів: передпроектний, проєктний (етапи технічного та робочого проєктування), впровадження, супровід і аналіз функціонування;
- визначені завдання, вимоги та обґрунтування розробки та розроблено технічне завдання на розробку інформаційно-довідкової системи навчального закладу "Дніпровська загальноосвітня школа I - III ступенів №5".

Під час дослідження встановлено, що для забезпечення централізованого сховища даних на серверній частині та паралельної багатокористувацької обробку даних база даних повинна мати клієнт-серверну архітектуру.

## РОЗДІЛ 2.

# ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ БАЗИ ДАНИХ ІНФОРМАЦІНО-ДОВІДКОВОЇ СИСТЕМИ

### 2.1. Клієнт – серверна архітектура СУБД

Застосування локальних СУБД мали певні обмеження, що не сприяли створенню ефективних додатків. Локальні СУБД не містять спеціальних додатків і сервісів, що можуть керувати даними, а використовують для цієї мети файлові сервіси операційної системи. Вся реальна обробка даних у таких СУБД здійснюється в клієнтському додатку, і будь які бібліотеки доступу до даних в цьому випадку також знаходяться в адресному просторі клієнтського додатку. Тому при виконанні запитів у таких СУБД, дані повинні бути доставлені в той же самий адресний простір клієнтського додатку. Це і призводить до перевантаження мережі при збільшенні числа користувачів і обсягу даних, а також загрожує іншими неприємними наслідками, наприклад, руйнуванням індексів і таблиць. Тому на сучасному етапі для створення додатків застосовуються клієнт-серверні СУБД [34, с. 268].

Архітектура клієнт-сервер є однією з домінуючих концепцій у створенні розподілених мережних додатків і передбачає взаємодію та обмін даними в всередині мережі.

*Технологія клієнт-сервер* означає такий спосіб взаємодії програмних компонентів, при якому вони утворюють єдину систему. Як видно з назви, існує клієнтський процес, що вимагає певних ресурсів, а також серверний процес, що ці ресурси надає. Зовсім необов'язково, щоб вони перебували на одному комп'ютері. Звичайно прийнято розміщувати сервер на одному вузлу локальної мережі, а клієнтів - на інших вузлах [30, с.15].

У контексті бази даних клієнт управляє користувацьким інтерфейсом і логікою роботи, діючи як робоча станція. Клієнт приймає від користувача запит, перевіряє синтаксис і генерує запит до бази даних мовою SQL або іншою мовою бази даних, відповідно до логіки роботи програми-клієнта. Потім передає повідомлення серверу, очікує надходження відповіді й форматує отримані дані для подання їх користувачеві. Сервер приймає й опрацьовує

запити до бази даних, після чого відправляє отримані результати назад клієнтові. Таке опрацювання включає перевірку повноважень клієнта, забезпечення вимог цілісності, а також виконання запиту й оновлення даних. Крім цього підтримується управління паралельністю й відновленням.

Архітектура клієнт-сервер має ряд переваг [4, с.1]:

- забезпечується ширший доступ до існуючих баз даних;
- підвищується загальна продуктивність системи: оскільки клієнти та сервер перебувають на різних комп'ютерах, їхні процесори можуть виконувати різні завдання паралельно. Налаштування продуктивності комп'ютера із сервером спрощуються, якщо на ньому виконується тільки робота з базою даних;
- знижується вартість апаратного забезпечення; досить потужний комп'ютер з більшим пристроєм зберігання потрібний тільки серверу – для зберігання й управління базою даних;
- скорочуються комунікаційні витрати. Частина операцій виконують клієнтські комп'ютери і посилають через мережу тільки запити до баз даних, що дозволяє значно скоротити обсяг даних, що пересилають мережею;
- підвищується рівень несуперечливості даних, оскільки кожній клієнтській програмі не доведеться виконувати власну перевірку їх цілісності;
- архітектура клієнт-сервер природно відображається на архітектуру відкритих систем.

У процесі розвитку технології клієнт-сервер змінювалися і способи її реалізації. Подальше розширення дворівневої архітектури клієнт-сервер припускає поділ функціональної частини колишнього, «товстого» (інтелектуального) клієнта на дві частини.

Способи організації доступу до даних за технологією клієнт-сервер та відмітимо деякі особливості кожного варіанту наведені нижче [34, с.320].

***Дворівнева архітектура клієнт-сервер.*** СУБД поділяється на дві частини: клієнтську й серверну. У цій архітектурі на виділеному сервері, що

працює, як правило, під управлінням серверної операційної системи, встановлюють спеціальне програмне забезпечення (ПЗ) - сервер БД. Основа роботи сервера БД - використання мови запитів (SQL). Запит мовою SQL, переданий клієнтом (робочою станцією) серверу БД, породжує пошук і вибір даних на сервері. Вибрані дані транспортуються мережею від сервера до клієнта (див. рис. 2.1).

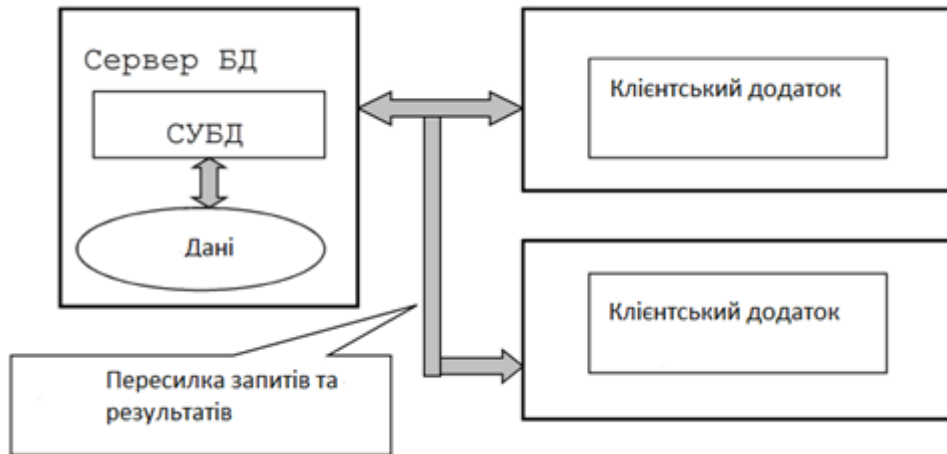


Рис. 2.1. Дворівнева архітектура «клієнт-сервер»

Виділяють такі особливості використання дворівневої архітектури:

- можливість використання різноманітних засобів, що надає СУБД, завдяки безпосередньому з'єднанню з сервером БД;
- значно знижується навантаження на мережу. Завдяки відправленню запитів і опрацюванню їх на сервері клієнт обмінюється із сервером тільки необхідними даними;
- складніше піддається масштабуванню, ніж трирівнева архітектура, тому що доводиться використовувати різні клієнти для різних операційних систем;
- потребує додаткових затрат для облаштування (встановлення) клієнтської частини;
- при модифікаціях серверної частини (наприклад додаванні нових функцій, зміні базової СУБД тощо) як правило потребує заміни (поновлення) клієнтської частини.

**Трирівнева архітектура клієнт-сервер** функціонує в Intranet та Internet мережах. Клієнтська частина («тонкий клієнт»), з якою працює користувач, є

Web-браузером або прикладною клієнтською програмою, що взаємодіє із Web-сервісами. Уся програмна логіка винесена на сервер застосувань, що забезпечує формування запитів до БД, які передаються серверу баз даних для їх виконання. Сервер застосувань може бути Web- сервером або спеціалізованою серверною програмою (наприклад, Oracle Forms Server). Схема такої структури подана на рис 2.2

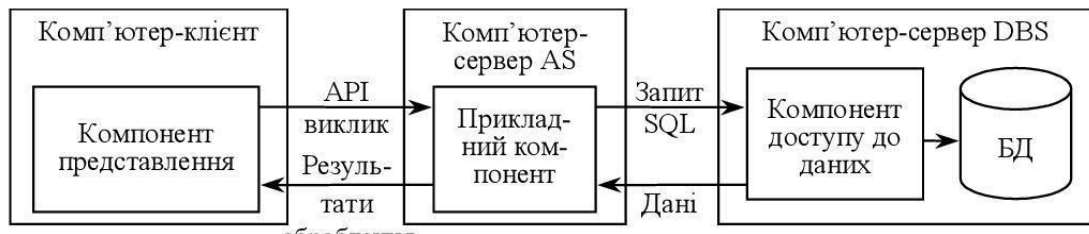


Рис. 2.2. Схема трирівневої архітектури «клієнт-сервер»

Саме трирівнева архітектура клієнт-сервер лежить в основі розробки сайтів, порталів. Наприклад, численні Internet-магазини, пошукові й довідкові сервери, системи для Internet-телефонії й обміну повідомленнями в реальному часі, системи передавання відео через Internet тощо.

Технологія "клієнт-сервер" стосовно до СУБД зводиться до поділу системи на дві частини - додаток-клієнт (Front-end) і сервер бази даних (back-end). Архітектура "клієнт-сервер" значно спрощує і прискорює розробку додатків за рахунок того, що правила перевірки цілісності даних знаходяться на сервері. Неправильно працюючий клієнтський додаток не може призвести до втрати даних. Всі ці можливості, раніше властиві тільки складним і дорогим системам, зараз доступні навіть невеликим організаціям.

Основа роботи сервера БД - використання мови запитів SQL (Structured Query Language). SQL-сервер забезпечує інтерпретацію запиту, його виконання в базі даних, формування результату виконання запиту і видачу його додатку-клієнту. При цьому ресурси клієнтського комп'ютера не беруть участь у фізичному виконанні запиту; клієнтський комп'ютер лише відсилає запит до серверної БД і отримує результат, після чого інтерпретує його необхідним чином і являє користувачу. Витягнуті дані транспортуються по мережі від сервера до клієнта. Тим самим, кількість переданої по мережі інформації зменшується в багато разів.

Отже, архітектуру клієнт-сервер, широко застосовують у сучасних додатках з застосуванням баз даних.

## **2.2. Порівняльний аналіз СУБД клієнт – серверної архітектури**

Клієнт-серверна СУБД – це система, що використовує технологію «клієнт-сервер». Такі СУБД складаються з клієнтської частини (яка входить до складу прикладної програми) і сервера. Клієнт-серверна СУБД дозволяє обмінюватися клієнту і сервера мінімально необхідними обсягами інформації. При цьому основна обчислювальна навантаження лягає на сервер. Клієнт може виконувати функції попередньої обробки перед передачею інформації сервера, але в основному його функції полягають в організації доступу користувача до сервера [36, с. 165].

На сучасному етапі існує багато СУБД, які мають клієнт-серверну архітектуру. Серед таких є Firebird, Interbase, MS SQL Server, Oracle, PostgreSQL, MySQL та інші. Кожна з них має свої переваги та недоліки.

Взагалі комерційні СУБД (Oracle, MS SQL Server та ін.) дорого коштують. Не кожне підприємство може придбати ліцензію на їх використання. Тому є доцільним застосовувати СУБД, які мають вільний код. Серед таких СУБД розглянемо найбільш популярні Firebird, PostgreSQL, MySQL.

**Firebird** – компактна, кроссплатформна, вільна система керування базами даних, що працює на Linux, Microsoft Windows і різноманітних Unix платформах [6, с. 22].

Як перевагу Firebird можна відзначити архітектуру, що має багато версій (паралельна обробка оперативних і аналітичних запитів: читають користувачі, що зчитують дані не блокують, тих хто записує), компактність (дистрибутив 10Mb), високу ефективність і потужну мовну підтримку для збережених процедур і тригерів.

Firebird використовується в різних промислових системах (складські та господарські, фінансовий і державний сектори) з 2001 р. Це комерційно незалежний проєкт у вигляді вільної версії Interbase 6.0.

Серверною частиною є файл fbserver.exe, який виконує функції суперсервера. Сервер Firebird - це програма, яка виконується на вузлу хоста в

мережі, і має доступ до клієнтів з порту комунікації. Вона обслуговує запити безлічі клієнтів до безлічі баз даних, суперсервери (Superserver) є багатопоточним процесом, який запускає новий потік для кожного клієнта, що з'єднується [6, с. 36].

Робота сервера включає: управління зберіганням даних БД; управління транзакціями; підтримку блокування і статистики для БД; обробку запитів на додавання, зміна або видалення рядків, підтримку поточних і застарілих версій записів; підтримку метаданих БД; обслуговування клієнтських запитів на отримання результуючих даних та виконання збережених процедур; маршрутизацію повідомлень для клієнтів; підтримку кешованих даних.

Клієнтську частину виконує бібліотека fbclient.dll. Клієнтська бібліотека надає протокол з'єднання і транспортний рівень, які клієнтський додаток використовує для зв'язку з сервером.

**MySQL** – вільна реляційна система управління базами даних. Розробку і підтримку MySQL здійснює корпорація Oracle. MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL- ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

MySQL працює на великій кількості платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS / 2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003, WinCE, Windows Vista і

Windows 7,8.

**PostgreSQL** – вільна об'єктно-реляційна система управління базами даних.

Існує в реалізаціях для безлічі UNIX-подібних платформ, включаючи AIX, різні BSD-системи, HP-UX, IRIX, Linux, Mac OS X, Solaris / OpenSolaris, Tru64, QNX, а також для Microsoft Windows. PostgreSQL базується на мові SQL і підтримує багато з можливостей стандарту SQL: 2003.

У таблиці 2.1. наведено порівняльний аналіз СУБД клієнт-серверної архітектури, що вільно розповсюджуються.

Таблиця 2.1

**Порівняльний аналіз СУБД клієнт-серверної архітектури**

<b>Критерії, за якими здійснюється аналіз</b>	<b>Firebird</b>	<b>MySQL</b>	<b>PostgreSQL</b>
Доступність – вільно розповсюджувана	Так	Так	Так
Багатоплатформна	Так	Так	Так
Максимальний розмір таблиці	-	4 Гб (ver.3.2)	32 Тб
Максимальний розмір бази даних	Обмежено файловою системою	Обмежено файловою системою	відсутні
Наявність багатоверсійності (MVCC)	Так	частково	Так
Тест продуктивності движків (багатопоточний запит)	0,005с	14,5 (MyISAM) 1,385(InnoDB)	3,1с

Таким чином, аналіз СУБД довів, що всі розглянуті програмні продукти мають позитивні риси, а саме: вільно розповсюджуються, мають кросплатформну спрямованість. Але багатоверсійність, як один з механізмів забезпечення одночасного конкурентного доступу до БД, СУБД MySQL підтримує частково: а саме тільки деякими засобами MySQL – Maria, InnoDB, Falcon, XtraDB. Багатопоточний тест продуктивності СУБД довів, що найбільш швидкісною є СУБД Firebird [5]. Тому саме цю СУБД беремо для розробки бази

даних клієнт-серверної архітектури.

### **2.3. Проєктування моделі бази даних інформаційно-довідкової системи**

Моделювання бази даних здійснюється на другому - проєктному етапі технології розробки інформаційно-довідкової системи.

Розробка бази даних вимагає її ретельного проєктування, а саме розробки структури (схеми) бази даних. Проєктування БД передбачає наступні етапи [24, с.7 – 11]:

1. Аналіз об'єкта автоматизації.
2. Інфологічне проєктування.
3. Даталогічне проєктування.
4. Фізичне проєктування.

1. Під час розробки технічного завдання до проєкту встановлено, що *об'єктом автоматизації* є навчального закладу "Лисичанська загальноосвітня школа I - III ступенів №5 Лисичанської міської ради Луганської області".

Інформаційно-довідкова система навчального закладу складається з інформації про навчальний процес: відомості про вчителя, учня, їх батьків, навчальний процес, відвідування занять учнями тощо.

2. *Інфологічне проєктування* передбачає створення інформаційної моделі, яка найбільш повно описує предметну галузь, без прив'язки до типу ЕОМ і типу системних програмних засобів [12, с. 112].

Однією з найбільш популярних засобів формалізованого представлення предметної галузі на етапі концептуального проєктування є ER- модель „сутність- зв'язок” (entity - relationship model). Послідовність проведення ER- моделювання має наступне:

- визначення типів сутностей;
- визначення типів зв'язків;
- визначення атрибутів сутностей і зв'язків;
- визначення потенційних і первинних ключів;
- перевірка моделі на відсутність надмірності.

Під час аналізу було визначено наступні сутності та їх характеристики:

а) *Вчитель* – містить дані про вчителів школи: ПІБ, дата народження,

стать, паспортні дані, адреса проживання, ідентифікаційний код, документ про освіту, кваліфікація, категорія, адміністративна посада, дата прийому та звільнення з роботи.

б) *Предмет* – дання про навчальні предмети, що викладаються учням: назва предмету, кількість годин, що відводиться на викладання, клас, де викладається цей предмет.

с) *Кваліфікація* – відомості про кваліфікацію вчителів.

д) *Учень* – містить наступну інформацію: номер особової справи учня, ПІБ, дата народження, стать, адреса проживання, національність, свідоцтво про народження, клас, де навчається, дата прийому та відрахування зі школи.

е) *Навчальний процес* – відомості про учня, предмет та оцінки.

ф) *Національність* – відомості про національність.

г) *Стан здоров'я* – дані про групу здоров'я, де учень займається фізичним вихованням, довідка з медустанови про стан здоров'я, дата початку та закінчення дозволу на звільнення від занять з фізкультури.

h) *Батьки* – дані про батьків учня: ПІБ батька, матері, опікуна (за наявності), кількість дітей у сім'ї, місце праці батька та матері.

і) *Факультатив* – відомості про факультатив, учнів що вивчають цей факультатив, вчителів, що викладають.

j) *Навчальний заклад* – відомості про вищий навчальний заклад, що закінчив вчитель.

к) *Клас* – дані про клас, кількість учнів та класного керівника.

l) *Пропуски занять* – інформація про кількість пропущених годин занять учнем за кожен місяць.

Також було встановлено атрибути сутностей, що є набором певних властивостей (табл. 2.2). Кожен атрибут був проаналізований на унікальність та можливість визначення у якості первинного або потенційного ключа, також визначені зовнішні ключі, що допоможуть встановити певні зв'язки між цими сутностями.

## Сутності та атрибути предметної галузі

№	Сутність	Набір атрибутів та їх характеристика
1	Вчитель	Ідент_вчитель (первинний ключ), прізвище, ім'я, по_батькові, дата_народження, стать, серія_паспорт, номер_паспорт, ким_виданий_паспорт, адреса_проживання, ідент_налоговий_код (потенційний ключ), категорія, серія_диплом, номер_диплом, ідент_навчальний_заклад (зовнішній ключ), ідент_кваліфікація (зовнішній ключ), амін_посада, дата_прийом, дата_звільнення.
2	Предмет	Ідент_предмет (первинний ключ), ідент_вчитель (зовнішній ключ), ідент_клас (зовнішній ключ), назва_предмет, години_предмет.
3	Кваліфікація	Ідент_кваліфікація (первинний ключ), назва_кваліфікація.
4	Учень	Іден_учень (первинний ключ), номер_особ_справа (потенційний ключ), прізвище, ім'я, по_батькові, дата_народження, стать, адреса_проживання, ідент_національність (зовнішній ключ), документ_особи, серія_документ, номер_документ, ідент_сім'я (зовнішній ключ), ідент_клас (зовнішній ключ), індив_навчання, дата_зарахування, дата_відрахування.
5	Навчальний процес	Ідент_навчальний_процес (первинний ключ), ідент_учень (зовнішній ключ), ідент_предмет (зовнішній ключ), оцінка_1_семестр, оцінка_2_семестр, оцінка_рік.
6	Національність	Ідент_національність (первинний ключ), назва_національність.
7	Стан здоров'я	Ідент_здоров'я (первинний ключ), ідент_учень (зовнішній ключ), група_здоров'я, довідка_здоров'я, дата_початок, дата_закінчення.
8	Батьки	Ідент_сім'я (первинний ключ), ПІБ_батька, ПІБ_мати, опікун, кількість_дітей, праця_батько, праця_мати.

№	Сутність	Набір атрибутів та їх характеристика
9	Факультатив	Ідент_факультатив (первинний ключ), назва_факультатив, ідент_учень (зовнішній ключ), ідент_вчитель (зовнішній ключ), кількість_годин
10	Навчальний заклад	Ідент_навчальний_заклад (первинний ключ), назва_навчальний_заклад, адреса_навчальний_заклад
11	Клас	Ідент_клас (первинний ключ), номер_клас, буква_клас, кількість_учнів, ідент_вчитель (зовнішній ключ)
12	Пропуски занять	Ідент_пропуски (первинний ключ), ідент_учень (зовнішній ключ), місяць, кількість_годин.

За допомогою первинних та зовнішніх ключів було становлено зв'язки між сутностями. Ці зв'язки демонструють семантику взаємодії сутностей між собою. Встановлено 14 бінарних зав'язків між сутностями з показником кардинальності «один до багатьох» (1:M). Характеристика зв'язків наведена у табл. 2.3.

Таблиця 2.3

#### Характеристика зв'язків між сутностями

№	Назва зв'язку	Тип	Ім'я батьківської таблиці	Ім'я дочірньої таблиці
1	Національність	1:M	Національність	Учень
2	Пропуск	1:M	Учень	Пропуски занять
3	Зв'язки	1:M	Батьки	Учень
4	Стан здоров'я	1:M	Учень	Стан здоров'я
5	Результат	1:M	Учень	Навчальний процес
6	Навчається1	1:M	Клас	Учень
7	Навчається2	1:M	Учень	Факультатив
8	Викладається	1:M	Клас	Предмет
9	Оцінка	1:M	Предмет	Навчальний процес
10	Викладає1	1:M	Вчитель	Предмет
11	Викладає2	1:M	Вчитель	Факультатив

12	Керівник	1:M	Вчитель	Клас
13	ВНЗ	1:M	Навчальний заклад	Вчитель
14	Кваліфікація	1:M	Кваліфікація	Вчитель

Наступним етапом було проведено аналіз на надмірність база даних. Встановлено, що зв'язки типу «один до одного», які відповідають в моделі одному й тому ж концептуальному об'єкту, відсутні. Надлишкових зв'язків теж не встановлено. Також, аналіз довів, що наведена схема даних знаходиться у третій нормальній формі (3НФ), що дозволяє зробити висновок про відсутність надмірності й суперечності БД.

Результатом інфологічного моделювання є побудова графічного зображення ER- моделі відповідно до нотацій Чена, яка подана у додатку А. Наступний етап проєктування бази даних - створення *дatalogічної моделі*, де проєктується структура даних і будується дatalogічна модель БД (схема БД), яка є описом логічної структури БД з урахуванням конкретної обраної СУБД (допустимі типи даних, найменування полів і таблиць, обмеження цілісності і т.п.) [24, с. 10].

У попередньому розділі було обґрунтовано вибір СУБД клієнт-серверної архітектури, тому для подальшого проєктування буде застосовуватися FireBird ver 2.5.3 32 bit для операційної системи Windows.

Аналіз атрибутів з точки зору дatalogічного моделювання наведений у таблицях 2.4-2.15.

Таблиця 2.4

### Сутність Учень (Pupils)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язковий Так\ ні	За замовчуванням
Іден_учень	P_id_pupil	Integer	первинний		Так	
номер_особ_справа	P_case_number	Varchar		16	Так	
прізвище	P_last_name	varchar		32	Так	
ім'я	P_first_name	Varchar		32	Так	
по_батькові	P_parent_name	Varchar		32	Так	

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язковий Так\ні	За замовчуванням
дата_народження	P_birth_date	Date			Ні	
стать	P_sex	Varchar		10	Ні	чол
адреса_проживання	P_address	Varchar		128	Ні	
ідент_національність	P_id_nationality	integer	зовнішній		Так	
документ_особи	P_document_type	Varchar		16	Ні	свідоцтво
серія_документ	P_doc_series	Varchar		16	Ні	
номер_документ	P_doc_number	Varchar		16	Ні	
ідент_сім'я	P_id_family	Integer	зовнішній		Так	
ідент_клас	P_id_class	Integer	зовнішній		Так	
індив_навчання	p_individual	Varchar		4	Так	ні
дата_зарахування	P_beginning_date	Date			Ні	
дата_відрахування	P_ending_date	date			Ні	01.01.1900

Таблиця 2.5

### Сутність Вчитель (Teachers)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язковий Так\ні	За замовчуванням
Ідент_вчитель	T_id_teacher	integer	первинний		Так	
прізвище	T_last_name	varchar		32	Так	
ім'я	T_first_name	varchar		32	Так	
по_батькові	T_parent_name	varchar		32	Так	
дата_народження	T_birth_date	date			Так	
стать	T_sex	varchar		10	Так	чол
серія_паспорт	T_doc_series	varchar		16	Так	
номер_паспорт	T_doc_number	varchar		16	Так	
кім_виданий_паспорт	T_doc_issue	varchar		64	Так	
адреса_проживання	T_address	varchar		128	ні	
ідент_налоговий_код	T_identification_code	varchar		16	ні	
категорія	T_category	varchar		16	Так	спеціаліст
серія_диплом	T_degree_series	varchar		16	ні	
номер_диплом	T_degree_number	varchar		16	ні	

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язково Так\ні	За замовчуванням
ідент_навчальний_заклад	T_id_education	integer	зовнішній		Так	
ідент_кваліфікація	T_id_qualification	integer	зовнішній		Так	
амін_посада	T_position	varchar		32	ні	01.01.1900
дата_прийому	T_start_of_work	Date			ні	
дата_звільнення	T_end_of_work	date			ні	

Таблиця 2.6

### Сутність Кваліфікація (Qualification)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язково Так\ні	За замовчуванням
Ідент_кваліфікація	Q_id_qualification	integer	первинний		Так	
назва_кваліфікація	Q_name	varchar		50	Так	

Таблиця 2.7

### Сутність Предмет (Subject)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язково Так\ні	За замовчуванням
Ідент_предмет	S_id_subject	Integer	первинний		Так	
ідент_вчитель	S_id_teacher	integer	зовнішній		Так	
ідент_клас	S_id_class	Integer	зовнішній		Так	
назва_предмет	S_name	Varchar		32	Так	
години_предмет	S_hours	integer			Так	0

Таблиця 2.8

### Сутність Навчальний процес (Progress)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній	Розмір	Обов'язково Так\ні	За замовчуванням
Ідент_навчальний_процес	P_id_progress	integer	первинний		Так	
ідент_учень	P_id_pupil	Integer	зовнішній		Так	
ідент_предмет	P_id_subject	Integer	зовнішній		Так	
оцінка_1_семестр	P_first_semester_mark	Integer			Так	0
оцінка_2_семестр	P_second_semester_mark	Integer			Так	0
оцінка_рік	P_year_mark	float			ні	

**Сутність Національність (Nationality)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/зовнішній ключ	Розмір	Обов'язкове Так\ ні	За замовчуванням
Ідент_національність	N_id_nationality	integer	первинний		Так	
назва_національність	N_nation	varchar	зовнішній	32	Так	

Таблиця 2.10

**Сутність Стан здоров'я (Health)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/зовнішній ключ	Розмір	Обов'язкове	За замовчуванням
Ідент_здоров'я	H_id_health	Integer	первинний		так	
ідент_учень	H_id_pupil	Integer	зовнішній		Так	
група_здоров'я	H_group	Varchar		64	Так	
довідка_здоров'я	H_doc_health	Varchar		128	Так	
дата_початок	H_begin_date_liberation	date			Так	
дата_закінчення	H_end_date_liberation	date			ні	

Таблиця 2.11

**Сутність Батьки (Family)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/зовнішній ключ	Розмір	Обов'язкове	За замовчуванням
Ідент_сім'я	F_id_family	Integer	первинний		так	
ПІБ_батька	F_father	Varchar		128	Ні	
ПІБ_мати	F_mather	Varchar		128	Ні	
опікун	F_guardian	Varchar		128	Ні	
кількість_дітей	F_child_count	Smallint			так	1
праця_батько	F_father_work	Varchar		128	Ні	
праця_мати	F_mather_work	Varchar		128	ні	

Таблиця 2.12

**Сутність Факультатив (Elective)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язко ве Так\ ні	За замовчува нням
Ідент_факультатив	E_id_elective	Integer	Первинний		Так	
назва_факультатив	E_name	varchar		25	так	
ідент_учень	E_id_pupil	Integer	Зовнішній		Так	
ідент_вчитель	E_id_teacher	Integer	зовнішній		Так	
кількість_годин	E_hours	Integer			Так	

Таблиця 2.13

**Сутність Навчальний заклад (Education)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язк о ве Так\ ні	За замовчув а нням
Ідент_навчальний_заклад	E_id_education	Integer	первинний		Так	
назва_навчальний_заклад	E_name	Varcha r		64	Так	
адреса_навчальний_закла д	E_address	varchar		128	так	

Таблиця 2.14

**Сутність Клас (Class)**

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язко ве Так\ ні	За замовчува нням
Ідент клас	C_id_class	Integer	первинний		Так	
номер клас	C_level	Integer			Так	
буква клас	C_name	Char		2	Так	
кількість_учнів	C_class_count	integer			Так	
ідент вчитель	C_id_teacher	Integer	зовнішній		так	

## Сутність Пропуски занять (Absence)

Назва атрибута	Назва поля у FireBird	Тип	Первинний/ зовнішній ключ	Розмір	Обов'язкове Так\ ні	За замовчуванням
Ідент_пропуски	A_id_absence	Integer	первинний		Так	
ідент_учень	A_id_pupil	Integer	зовнішній		Так	
місяць	A_manth	Varchar		16	Так	
кількість_годин	A_hours	integer			Так	

Даталогічна модель наведена на рис. 2.3

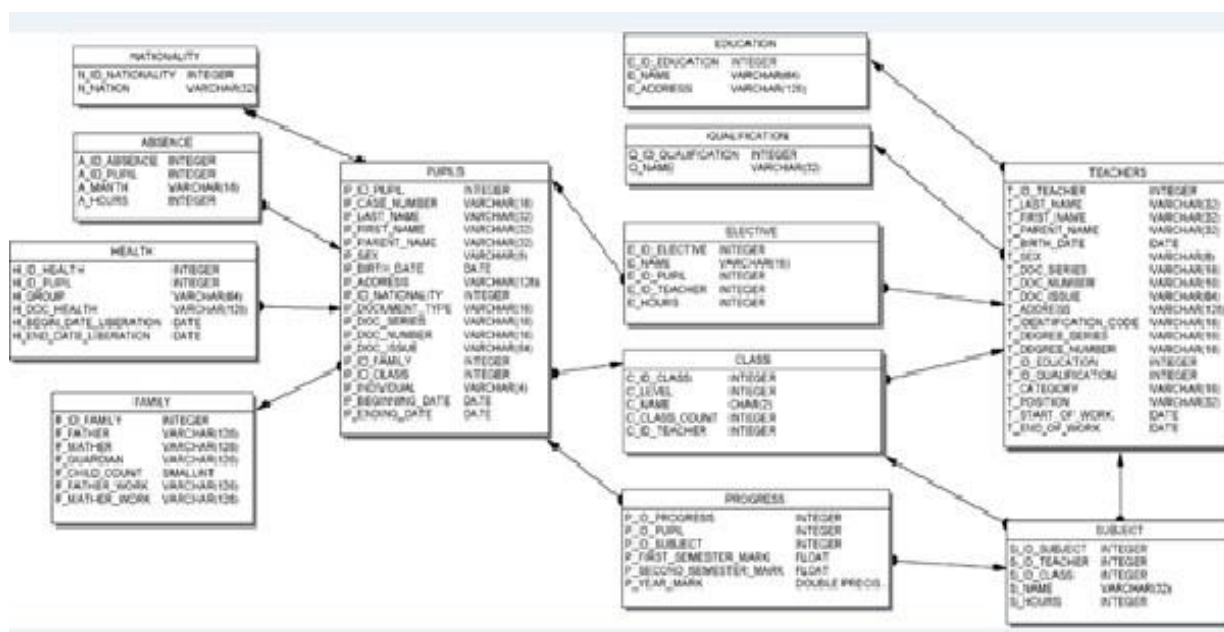
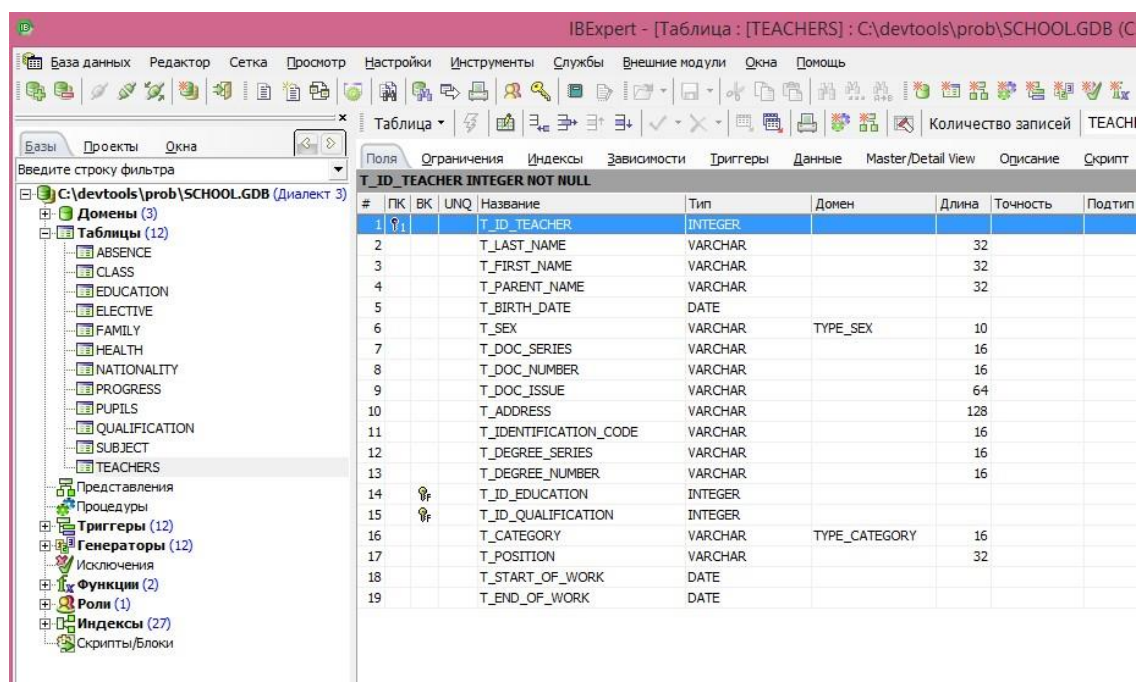


Рис. 2.3. Даталогічна модель БД

Після проєктування бази даних здійснюється **фізична реалізація** моделі БД, яка складається зі створення та завантаження даних в БД, розробку і налагодження прикладних програм для роботи з базою даних, оформлення документації. На цьому етапі будується **фізична модель БД**, яка описує способи фізичної організації даних.

Для зручного проєктування скористуємося програмним засобом IVExpert - GUI-оболонкою, призначеною для розробки та адміністрування баз даних Firebird, а також для вибору та зміни даних, що зберігаються в базах. Для роботи з цим програмним засобом, в-першу чергу, потрібно встановити сервер бази даних FireBird та запустити його на виконання. Після встановлення IVExpert потрібно зареєструвати нову базу даних: користувач SYSDBA, пароль –

masterkey. Потім – під’єднатися до цієї бази. Приклад створеної таблиці наведено на рис. 2.4



#	PK	BK	UNQ	Название	Тип	Домен	Длина	Точность	Подтип
1	1			T_ID_TEACHER	INTEGER				
2				T_LAST_NAME	VARCHAR		32		
3				T_FIRST_NAME	VARCHAR		32		
4				T_PARENT_NAME	VARCHAR		32		
5				T_BIRTH_DATE	DATE				
6				T_SEX	VARCHAR	TYPE_SEX	10		
7				T_DOC_SERIES	VARCHAR		16		
8				T_DOC_NUMBER	VARCHAR		16		
9				T_DOC_ISSUE	VARCHAR		64		
10				T_ADDRESS	VARCHAR		128		
11				T_IDENTIFICATION_CODE	VARCHAR		16		
12				T_DEGREE_SERIES	VARCHAR		16		
13				T_DEGREE_NUMBER	VARCHAR		16		
14				T_ID_EDUCATION	INTEGER				
15				T_ID_QUALIFICATION	INTEGER				
16				T_CATEGORY	VARCHAR	TYPE_CATEGORY	16		
17				T_POSITION	VARCHAR		32		
18				T_START_OF_WORK	DATE				
19				T_END_OF_WORK	DATE				

Рис. 2.4. Приклад таблиці Teachers

Аналогічно створюються всі дванадцять таблиць. Потім за допомогою встановлення зовнішніх ключів визначаються зв'язки між таблицями.

Таким чином, на цьому етапі було розроблено інфологічну, даталогічну модель даних на підставі яких було розроблено фізичну модель бази даних school.gdb.

### Висновки до розділу

У другому розділі було спроектовано та розроблено базу даних системи інформаційно-довідкової системи навчального закладу. Під час цієї діяльності було досягнуто наступних результатів:

- розглянуто теоретичні підходи до технології клієнт-серверної архітектури. Встановлено, що використання клієнт-серверних СУБД є більш ефективними в порівнянні з локальними базами даних, завдяки забезпеченню широкого доступу до існуючих БД, підвищенню модульності програмного продукту;
- проаналізовано сучасні клієнт-серверні СУБД, які вільно розповсюджуються. Під час аналізу було визначено СУБД для розробки інформаційно-довідкової системи навчального закладу –

FireBird, яка дозволяє працювати з реляційними базами даних й вирішувати встановлені в технічному завданні задачі;

- під час методологічного проєктування визначено дванадцять сутностей, їх атрибути та зв'язки між ними, також розроблено інфологічну та даталогічну модель бази даних інформаційно-довідкової системи навчального закладу;
- за допомогою засобу адміністрування баз даних IBExpert розроблено фізичну модель бази даних school.gdb.

## **РОЗДІЛ 3.**

### **РОЗРОБКА ДОДАТКУ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ**

#### **3.1. Аналіз засобів та технологій розробки додатку**

Наступним шагом технології розробки ІДС є розробка програмного забезпечення (ПЗ) — це процес, направлений на створення і підтримку працездатності, якості і надійності програмного забезпечення, використовуючи технології, методологію і практики з інформатики, управління проектами, математики, інженерії і інших областей знання. Тому дуже важливо підійти до вибору інструментів та технологій розробки ПО.

При виборі мови програмування були розглянуті дві мови – C++ і C#. У наведених нижче таблицях 3.1–3.9. відмічена наявність або відсутність тих чи інших можливостей в розглянутих мовах програмування C++ і C# [39]. В таблицях було зроблено порівняння з таких характеристик: парадигма, типізація, компілятор/інтерпретатор, управління пам'яттю, управління потоком обчислень, типи та структури даних, об'єктно-орієнтовані можливості та інші можливості.

Умовні позначення в таблицях:

+ можливість існує

- можливість відсутня

+/- можливість підтримується не повністю

-/+ можливість підтримується дуже обмежено

Таблиця 3.1

## Парадигми

Мови програмування	Можливості							
	Імперативна	Об'єктно-орієнтована	Функціональна	Рефлексивна	Узагальнене програмування	Логічна	Декларативна	Розподілена
C++	+	+	+/-	-	+	-	-	+/-
C#	+	+	+/-	-/+	+	-	-/+	-/+

Таблиця 3.2

## Типізація

Можливості	C++	C#
Статична типізація	+	+
Динамічна типізація	-	+
Явна типізація	+	+
Неявна типізація	-/+	-/+
Неявне приведення типів без втрати даних	+	+
Неявне приведення типів з втратою даних	+	-
Неявне приведення типів в неоднозначних ситуаціях	+	+
Аліаси типів	+	+
Вивід типів змінних з ініціалізатора	+/-	+
Вивід типів змінних з використання	+/-	+
Вивід типів - аргументів при виклику методу	+	+
Вивід сигнатури для локальних функцій	+/-	-

Таблиця 3.3

## Керування пам'яттю

Мови програмування	Можливості			
	Створення об'єктів на стеку	Некеровані вказівники	Ручне керування пам'яттю	Збірка сміття
C++	+	+	+	-/+
C#	+	+	+	+

Таблиця 3.4

## Компілятор/інтерпретатор

Мови програмування	Можливості					
	Open-source компілятор (інтерпретатор)	Можливість компіляції	Bootstrapping	Багатопоточна компіляція	Інтерпретатор командного рядка	Умовна компіляція
C++	+	+	+	+	+/-	+
C#	+	+	+	-	-	+

Таблиця 3.5

## Керування потоком обчислень

Мови програмування	Можливості						
	Інструкція goto	Інструкції break без мітки	Інструкція break з міткою	Підтримка try / catch	Блок finally	Блок else (виключення)	Ледачі обчислення
C++	+	+	-	+	-	-	-/+
C#	+	+	-	+	+	+	-/+

Таблиця 3.6

## Типи та структури даних

Мови програмування	Можливості							
	Кортежі	Багатомірні масиви	Динамічні масиви	Асоціативні масиви	Контроль меж масивів	Цикл foreach	Спискові вклучення	Цілі числа довільної довжини
C++	+/-	+	+	+	+/-	+	-	-
C#	+/-	+	+/-	+	+	+	-/+	+

Таблиця 3.7

## Об'єктно-орієнтовані можливості

Мови програмування	Можливості				
	Інтерфейси	Мультиметоди	Mixins	Перейменування членів при спадкуванні	Множинне спадкування
C++	+	-/+	-/+	-/+	+
C#	+	-/+	-	-	-

## Функціональні можливості

Мови програмування	Можливості			
	Декларації чистоти функцій	First class functions	Анонімні функції	Лексичні замикання
C++	-	+	+	+
C#	-	+	+	+

Таблиця 3.9

## Різні можливості

Мови програмування	Можливості									
	Макроси	Шаблони / Generics	Підтримка Unicode в ідентифікаторах	Перевантаження функцій	Іменовані параметри	Значення параметрів за замовчуванням	Локальні функції	Зіставлення із	Контракне програмування	Наявність бібліотек для роботи з графікою і мультимедіа (OpenGL / WebGL / OpenML, OpenAL, ...)
C++	+	+	+	+	-	+	+	-	-	+
C#	-	+	+	+	+	+	+/-	+/-	+	+

Проведений аналіз мов програмування свідчить, що мови програмування C++ та C# є дуже потужними засобами створення програмних додатків. Можливості цих мов дуже схожі, але є ряд відмінностей. Фундаментальні основи переваг C++ в можливості писати код, який буде виконуватися безпосередньо процесором, і можливості прямої роботи з пам'яттю. Але в нашому проєкті це не принципово. Перевагою C# є те, що під Windows, C#-проєкти швидше розроблювати та помітно зручніше налагоджувати.

Крім того, одною з найбільш актуальних мов програмування, на якій

ведеться розробка клієнтських додатків для баз даних з використанням технології ADO.NET, є C # [37, с. 55]. Ця мова найбільш оптимізована розробником Microsoft для роботи з цією технологією.

Саме це обумовлює вибір C # як інструменту розробки інформаційно-довідкової системи навчального закладу, де однією з основних частин є клієнт-серверна база даних.

В технології Microsoft.NET створена достатня гнучка і ефективна модель доступу до даних - ADO.NET, що надає розробникам набір об'єктів, на основі яких можна створювати додатки будь-якого масштабу (від локальних до глобальних).

В цій технології закладена можливість роботи програми в стані "розриву" з'єднання з базою даних [37, с. 32]. Додатки підключаються до бази даних тільки на невеликий проміжок часу. З'єднання встановлюється тільки тоді, коли клієнт з віддаленого комп'ютера запрошує на сервері дані.

В об'єктній моделі ADO.NET можна виділити декілька класів (рис. 3.1) та кілька рівнів.

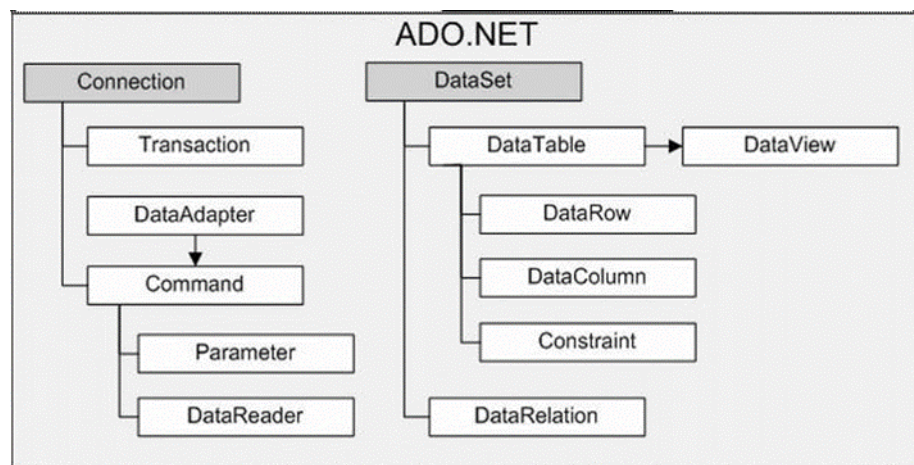


Рис. 3.1. Ієрархія класів ADO .Net

**Рівень даних.** Це базовий рівень, на якому розташовуються самі дані (наприклад, таблиці бази даних Firebird). На даному рівні забезпечується фізичне зберігання інформації на магнітних носіях і маніпуляція з даними на рівні вихідних таблиць (вибірка, сортування, додавання, видалення, оновлення).

**Рівень бізнес-логіки.** Це набір об'єктів, що визначають, з якою базою даних належить встановити зв'язок і які дії необхідно буде виконати з даними, що містяться в неї. Для встановлення зв'язку з базами даних використовується

об'єкт `FbDataConnection`. Для зберігання команд, що виконують будь-які дії над даними, використовується об'єкт `FbDataAdapter`. Для зберігання результатів вибірки використовується об'єкт `DataSet`.

**Рівень додатку.** Це набір об'єктів, що дозволяють зберігати і відображати дані на комп'ютері кінцевого користувача. Для зберігання інформації використовується об'єкт `DataSet`, а для відображення даних є досить великий набір елементів управління (`DataGrid`, `TextBox`, `ComboBox`, `Label` і т. Д.).

Таким чином, з урахуванням вибору мови програмування `C #` та технології доступу до даних `ADO.NET`, виберемо середовище `Visual Studio`.

### **3.2. Розробка та реалізація програмного додатку**

Під час розробки `C #`-додатків під `FireBird`, особливу увагу потрібно приділити підключенню до бази даних. Це можна зробити за допомогою компоненту `ADO.NET Data Provider`. Компонент `ADO.NET Data Provider` володіє всіма необхідними функціональними можливостями, які можуть бути затребувані при підключенні до `FireBird`. Компонент дозволяє змінювати параметри авторизації, підключення – дозволяє вибрати локальний або віддалений сервер, здійснювати налаштування порту і кодування, в якій буде здійснюватися робота з даними. Завантажити компонент можна на офіційному сайті `FireBird`. У проєкт додатку `Visual Studio` досить підключити файл `FirebirdSql.Data.FirebirdClient.dll`. Також в додаток необхідно додати рядок з вказаним простором імен:

```
using FirebirdSql.Data.FirebirdClient;
```

В додатку, що розроблюється, підключення до бази даних здійснюється за допомогою розробленого класу `FirebirdInterface` (додаток В). Цей клас розроблено з метою створення додаткового рівня абстракції над об'єктами `ADO.NET` і `ADO.NET Data Provider`. У таблиці 3.10 наведено основні методи, що реалізовані у цьому класі.

Основні методи, що реалізовані у класі FirebirdInterface

№	Ім'я методу	Призначення методу
1	InitConnectionString	Повертає сформований рядок з'єднання з базою даних на підставі переданих у функцію параметрів
2	openDataBase	Відкриває з'єднання з базою даних, у випадку успішного з'єднання функція повертає значення «істина»
3	closeDataBase	Закриває з'єднання з базою даних
4	isOpen()	Перевірка з'єднання з базою даних, у випадку відкритого з'єднання функція повертає значення «істина»
5	fillSchema	Функція ініціалізує властивості SelectCommand об'єкта FbDataAdapter
6	fill	Здійснює заливку даних з фізичної бази даних в об'єкт DataTable
7	save	Здійснює збереження змін у БД

В розробленому клієнтському додатку з'єднання з базою даних здійснюється у діалоговому вікні LoginDialog, що наведено у рис. 3.2.

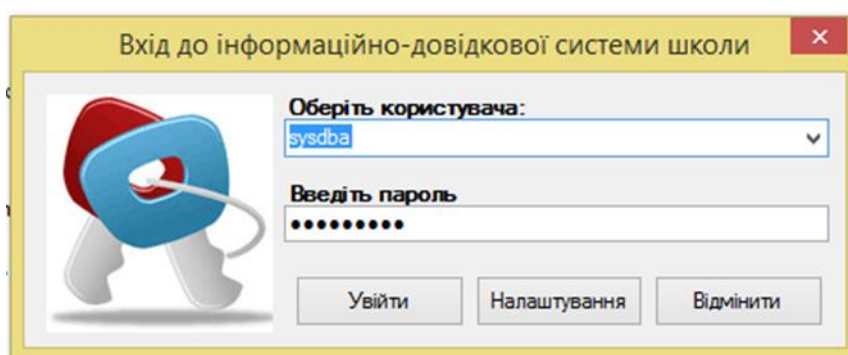


Рис. 3.2. Діалогове вікно LoginDialog

Підключення до бази даних здійснюється у методі Accept\_Click кнопки «Увійти». Код методу наведено нижче:

```
private void Accept_Click(object sender, EventArgs e)
{
```

```

        fbObject.InitConnectionString(ConnectItems.library,
                                      ConnectItems.database,
                                      ConnectItems.users[userList.SelectedIndex],
                                      password.Text,
                                      ConnectItems.role, ConnectItems.charset,
                                      ConnectItems.port);
        if (!fbObject.openDataBase()) result
            = DialogResult.Retry;
        else
            result = DialogResult.OK;
    }

```

В цьому методі формується рядок підключення до БД за допомогою виклику функції InitConnectionString:

```

    public string InitConnectionString(string clientLibrary, string database, string
    userID, string password, string role, string charset, int port)
    {
        FbConnectionStringBuilder connectString = new
FbConnectionStringBuilder();
        connectString.ClientLibrary = clientLibrary;
        connectString.Database = database;
        connectString.UserID = userID;
        connectString.Password = password;
        connectString.Charset = charset;
        connectString.Role = role; connectString.Port
        = port;
        connect.ConnectionString = connectString.ConnectionString;
        connect.InfoMessage += new
FbInfoMessageEventHandler(OnInfoMessage);
        return connect.ConnectionString;
    }

```

Функція openDataBase відкриває з'єднання з БД: public

```

bool openDataBase()
{
    try
    {
        if (connect.ConnectionString.Length > 0)
            connect.Open();
        else
            throw new System.Exception("Строка соединения с БД не
            корректна.");
    }
    catch (System.Exception Except)
    {
        MessageBox.Show(Except.Message, "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        connect.Close(); return
        false;
    }
    return connect.State == System.Data.ConnectionState.Open ?
    true : false;
}

```

Інтерфейс клієнтського додатку складається з головної форми, яка подана у рис 3.3.

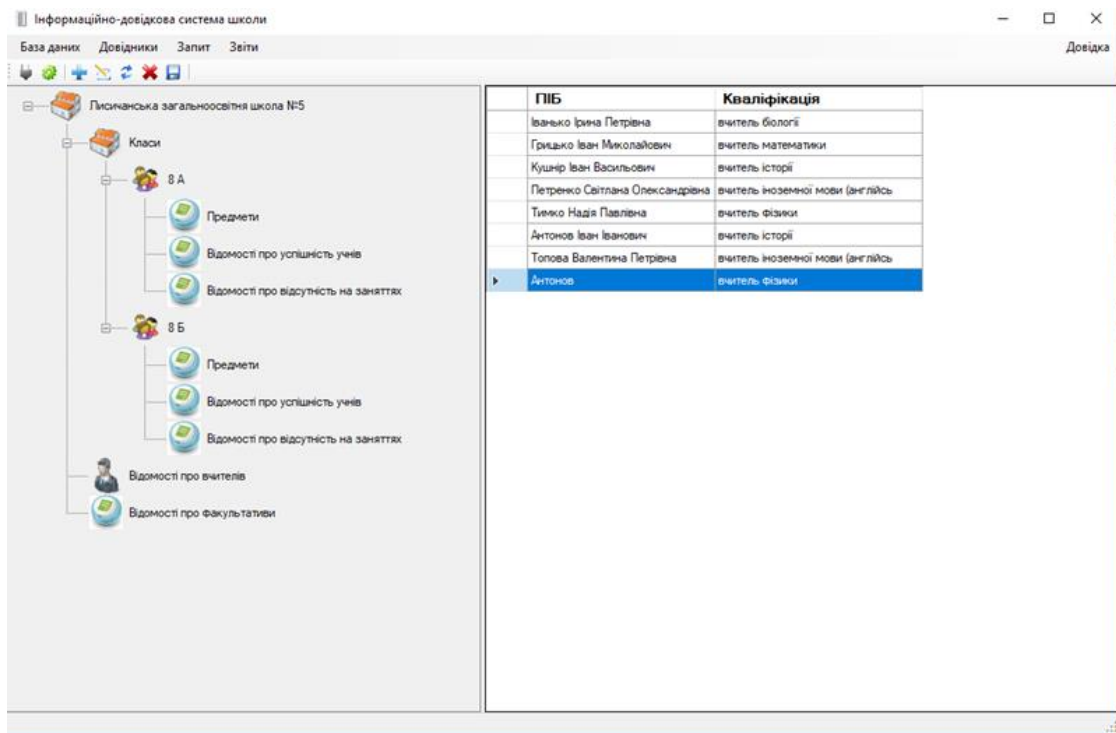


Рис 3.3. Головна форма клієнтського додатку системи

До складу форми входить головне меню та панель інструментів, з лівого боку розташовано ієрархічне деревовидне меню, з правого – головний елемент відображення змісту таблиць бази даних.

Головне меню складається з пунктів:

- «База даних» – налаштування, від'єднання від БД, вихід з системи;
- «Довідники» – інформація довідникового характеру, що викликає форми: гілка «Учень» – «Національність», «Відомості про батьків», «Стан здоров'я»; гілка «Вчитель» – «Вищий навчальний заклад», «Кваліфікація»;
- «Запити» – вибіркова інформація: гілка «Учень» – «Запит за ПІБ», «Успішність учня», «Відвідування занять учнем», «Запит за датою народження»; гілка «Вчитель» – «Запит за ПІБ», «Запит за категорією», «Навантаження викладача»;
- «Звіти» – інформація звітного характеру: «Відомість про склад учнів класу», «Картка навантаження вчителя», яку можна роздрукувати.

В головній формі відбиваються дані таблиць CLASS (класи), PUPILS (учні), SUBJECT (предмети), PROGRESS (успішність), ABSENCE (пропуски занять), ELECTIVE (факультативи).

При виборі назви класу в правій частині форми відображуються скороченні дані про учнів з таблиці PUPILS з ознакою приналежності до певного класу (за полем P\_ID\_CLASS). Повну інформацію про окремого учня можна отримати при натисканні мишею на ПІБ цього учня. При цьому відкривається форма «Учень», приклад якої подано у рис. 3.4. Ця форма має можливості не тільки відображення інформації про учня, а й додавання, видалення, оновлення даних.

Додавання нового учня здійснюється за допомогою кнопки «Додати запис», що розташована на панелі інструментів головної форми. При цьому відкривається аналогічна форма «Учень», але з порожніми полями.

Аналогічні принципи закладено у роботі з гілкою «Вчителі». Форма яка активується – «Особова картка вчителя».

The screenshot shows a window titled "Особова картка вчителя" (Teacher's Personal Card). It contains the following fields and controls:

- Дата народження** (Date of birth): 29 апреля 1968 г.
- ПІБ** (Full name): Тимко Надія Павлівна
- Ідентифікаційний номер** (Identification number): 123456789
- Адреса** (Address): М.Старобільськ, пл. Гоголя,1
- Серія паспорта** (Passport series): ЕК
- Номер паспорта** (Passport number): 123456
- Ким і коли виданий** (Issued by and when): Старобільським МВ УМВС України
- Серія диплому про освіту** (Education diploma series): ЕМ
- Номер диплому про освіту** (Education diploma number): 123456
- Назва навчального закладу, що закінчив** (Name of the educational institution completed): Бердянський державний педаг. +
- Посада** (Position): директор
- Кваліфікація** (Qualification): вчитель фізики +
- Категорія** (Category): вища
- Стать** (Gender): жін
- Дата початку праці** (Start date of work): 23 апреля 2001 г.
- Дата закінчення праці** (End date of work): 1 января 1900 г.
- Buttons**: Зберегти (Save), Відмінити (Cancel)

Рис. 3.4. Приклад форми «Особова картка вчителя»

Особливістю побудови головної форми є те, що об'єкт управління DataGridView має пристосовувати відображення даних в сітці в залежності від структури таблиць, що зв'язується з цим об'єктом. В об'єкті управління DataGridView повинні розміщуватися дані з різних зв'язаних таблиць, ховатися різні стовбці, бути різними заголовки стовбців. Для розв'язання цього завдання було застосовано поліморфізм, а саме створено базовий клас PolymorphicWidget, від якого створено похідні класу PupilsTable, SubjectTable, ProgressTable, AbsenceTable, TeacherTable, ElecativeTable, в яких перевизначається віртуальний метод adapt базового класу, що формує структуру відображуваної таблиці.

Приклад коду базового класу `PlymorphicWidget` й похідного від нього класу `SubjectTable` наведено нижче:

```
public class PlymorphicWidget
{
    protected TreeNode nodeData; protected
    MainWindow widget;
    public PlymorphicWidget(MainWindow window)
    {
        widget = window;
    }

    protected virtual void view(){} public
    virtual void adaptFilterA()
    {
        widget.mainGrid.DataSource = null;
    }

    public virtual void adaptFilterB()
    {
        widget.mainGrid.DataSource = null;
    }

    public virtual void adapt()
    {
        widget.mainGrid.DataSource = null;
    }

    public virtual void showEditDialog()
    {
    }

    public virtual void insertDialog()
    {
    }

    public void deleteRow()
    {
    }
```

```

if (widget.mainGrid.DataSource != null)
{
    if (DialogResult.Yes == MessageBox.Show("Видалити запис?", "Увага",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question))
    {
        DataRow row = widget.findCurrentRow(widget.mainGrid);
        if (row != null) row.Delete();
        else
            MessageBox.Show("Рядок не знайдено", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
else
    MessageBox.Show("Таблиця не визначена!", "Увага",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

public virtual void dataSave()
{
}

public virtual void dataRefresh()
{
}

public TreeNode Node
{
    get { return nodeData; }
    set { nodeData = value; }
}
}

```

Далі наведено перевизначений метод adapt похідного класу SubjectTable:

```

public class SubjectTable : PolymorphicWidget
{

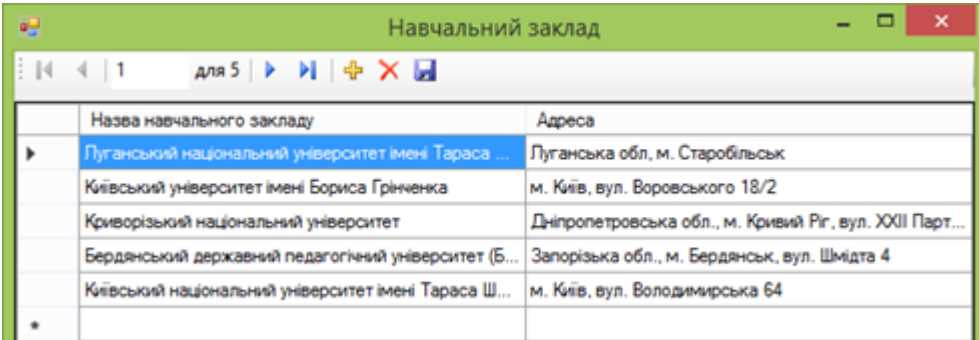
```

```

public override void adapt(MainWindow widget)
{
    widget.mainGrid.DataSource = 0;
    DataView view =
widget.subjectView[widget.treePosition[nodeData.Parent.Text]];
    widget.currentTable = view.Table.TableName;
    widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource
    = view; widget.initViewRelation("relation11", "Вчитель", 1, 5);
    widget.mainGrid.Columns[0].Visible = false;
    widget.mainGrid.Columns[1].Visible = false;
    widget.mainGrid.Columns[2].Visible = false;
    widget.mainGrid.Columns[3].HeaderText = "Назва предмету";
    widget.mainGrid.Columns[3].Width = 230;
    widget.mainGrid.Columns[4].HeaderText = "Кількість годин";
    widget.mainGrid.Columns[4].Width = 230;
}
public override void showDialog(MainWindow widget)
{
    MessageBox.Show("SubjectTable");
}
}

```

Приклад вікна довідника «Начальний заклад» наведено у рис. 3.5:



	Назва навчального закладу	Адреса
▶	Луганський національний університет імені Тараса ...	Луганська обл., м. Старобільськ
	Київський університет імені Бориса Грінченка	м. Київ, вул. Воровського 18/2
	Криворізький національний університет	Дніпропетровська обл., м. Кривий Ріг, вул. XXII Парт...
	Бердянський державний педагогічний університет (Б...	Запорізька обл., м. Бердянськ, вул. Шмідта 4
	Київський національний університет імені Тараса Ш...	м. Київ, вул. Володимирська 64
*		

Рис. 3.5. Приклад вікна довідника «Начальний заклад»

Навігація у вікні реалізована за допомогою bindingNavigator, до

стандартних компонентів якого було додано кнопку «Зберегти». Код методу при натисканні на кнопку наведено нижче:

```
private void saveToolStripButton_Click(object sender, EventArgs e)
{
    this.Validate();
    bindingSource_ed.EndEdit(); if
    (!fb.save("EDUCATION"))
    {
        MessageBox.Show("Збереження не виконано");
    }
}
```

В цьому методі викликається метод save об'єкта fbObject класу FirebirdInterface:

```
public bool save(string tableName)
{
    DataTable currentTable = dataTable(tableName);
    if (currentTable.GetChanges() == null) return false; try
    {
        tableAdapter(tableName).Update(currentTable.GetChanges());
        currentTable.AcceptChanges();
        return true;
    }
    catch (Exception)
    {
        currentTable.RejectChanges(); return
        false;
    }
}
```

У меню «Запити» здійснюється формування вибірки даних за параметром. Приклад вікна параметру для вибірки даних навантаження викладача за вказаним ПІБ наведено у рис. 3.6.

Рис. 3.6. Приклад вікна параметру ПІБ викладача

Результат формування вибірки даних за вказаним параметром наведено у рис. 3.7.

Вчитель	Клас	Назва предмета	Кількість годин
Грица Іван Миколайович	8 А	алгебра	45
Грица Іван Миколайович	8 Б	алгебра	45
Грица Іван Миколайович	8 А	геометрія	30
Грица Іван Миколайович	8 Б	геометрія	30

Рис. 3.7. Вибірка даних за параметром

Вибірка даних за параметром здійснюється за допомогою віртуальних методів, що перевизначаються: `adaptFilterA()` та `adaptFilterB()`. Приклад коду наведено нижче:

```
public override void adaptFilterA()
{
    widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject);
    if (widget.pupilsParamDlg.ShowDialog() == DialogResult.OK)
    {
        string filterParemetr = "P_NAME = " + "\"" +
            widget.pupilsParamDlg.cbPupilsName.Text + "\"";

        DataView view = new
            DataView(widget.fbObject.dataTable("PUPILS"), filterParemetr, "P_ID_PUPIL",
                DataViewRowState.CurrentRows);

        if (view.Count == 0)
```

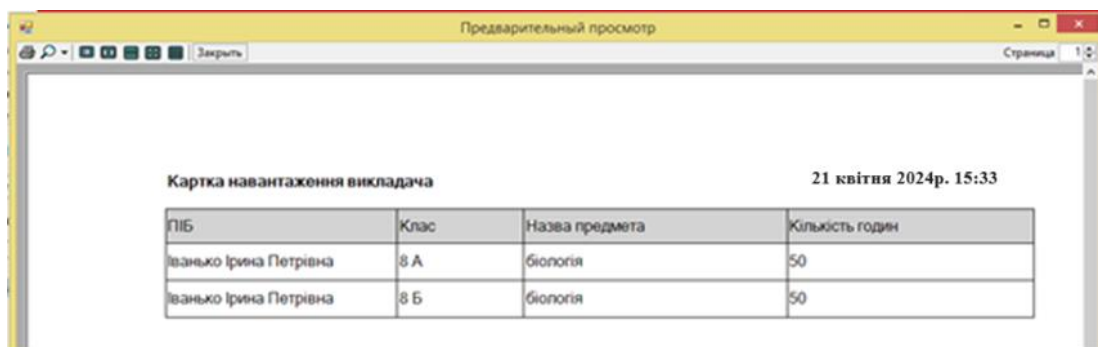
```

        MessageBox.Show("Дані відсутні в базі", "Увага!",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    else
    {
        widget.mainGrid.DataSource = 0; widget.currentTable
        = view.Table.TableName;
        widget.bindingSource1.DataSource =
        widget.fbObject.dataTable(widget.currentTable);
        widget.mainGrid.DataSource = view; this.view();
    }
}
}
}

```

У змінній `filterParemetr` формується параметр, за яким фільтруються дані, і який застосовується у `DataView`. Цей елемент застосовувався для фільтрації даних.

Меню «Звіти» формує документи на друк. На рис. 3.8 наведено приклад друку документу.



Карточка навантаження викладача				21 квітня 2024р. 15:33	
ПІБ	Клас	Назва предмета	Кількість годин		
Іванько Ірина Петрівна	8 А	біологія	50		
Іванько Ірина Петрівна	8 Б	біологія	50		

Рис. 3.8. Приклад форми друку

Після розробки клієнтського додатку було створено інсталяційний пакет, до складу якого входить сервер `FireBird`, скомпільований додаток, файл бази даних `school.gdb`.

Таким чином, було розроблено інформаційно-довідкову систему навчального закладу "Дніпровська загальноосвітня школа I - III ступенів №5" на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням.

### 3.3. Тестування додатка

#### *Об'єкт і цілі тестування*

Розроблюваний програмний комплекс повинен формувати інформаційну базу інформаційно-довідкової системи навчального закладу. Програмний проєкт повинен забезпечувати:

1. Введення (додавання) і відображення інформації з бази даних.
2. Пошук даних по введеному параметру.
3. Видалення, редагування даних;
4. Ведення довідників, які будуть використовуватися базою даних.
5. Формування звітних форм.

#### *Методи тестування*

В ході тестування будемо використовувати ручний метод. Ручне тестування (manual testing) - частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно проводиться тестувальниками або звичайними користувачі шляхом моделювання можливих сценаріїв дії користувача.

#### *Етапи тестування*

Тестування включало п'ять етапів, які відповідають цілям тестування.

1. Введення (додавання) і відображення інформації з бази даних.

Завдання: додати запис про предмети. В ході тестування заповнена форма «Відомості про предмети, що викладаються». Перегляд нового запису поданий з права (рис. 3.9).

The screenshot displays two windows from a school information system. The left window, titled 'Відомості про предмети, що викладаються', contains a form with the following fields: 'Назва предмету' (filled with 'Інформатика'), 'ПІБ викладача' (filled with 'Кушнір Іван Васильович'), 'Клас, в якому викладається предмет' (filled with '8 А'), and 'Кількість годин на рік' (filled with '60'). There are 'Зберегти' and 'Відмінити' buttons at the bottom. The right window, titled 'Інформаційно-довідкова система школи', shows a table of subjects.

Вчитель	Клас	Назва предмета	Кількість годин
Іваненко Ірина Петрівна	8 А	біологія	50
Грицько Іван Миколайович	8 А	алгебра	45
Грицько Іван Миколайович	8 А	геометрія	30
Кушнір Іван Васильович	8 А	історія	20
Тимко Надія Павлівна	8 А	фізика	40
Антонів Іван Іванович	8 А	іноземна мова	20
Кушнір Іван Васильович	8 А	інформатика	60

Рис. 3.9. Результати тестування введення (додавання) і відображення інформації в БД

Аналогічно проведено тестування на додавання та перегляд даних в інших

таблицях.

*Висновок:* операції введення і відображення даних в програмному проєкті відповідають вимогам.

2. Пошук даних за введеним параметром. Завдання: відфільтрувати таблицю Вчителя за параметром «Кушнір Іван Васильович», результат пошуку подана рис. 3.10.

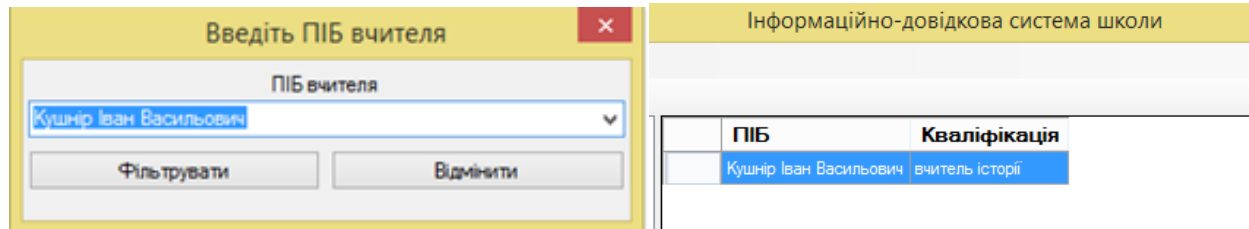


Рис. 3.10. Результати тестування пошуку даних за введеним параметром  
Аналогічно протестований інші пошуки за параметром.

*Висновок:* операції пошуку даних по введеному параметру в проєкті Програми відповідають вимогам.

3. Видалення, редагування даних. Завдання: видалити і оновити дані про вчителя. В ході тестування була виділена запис на видалення з прізвищем Кушнір Іван Васильович. Після натискання на панелі інструментів кнопки видалення з'являється діалогове вікно - запит на видалення запису (рис. 3.11).

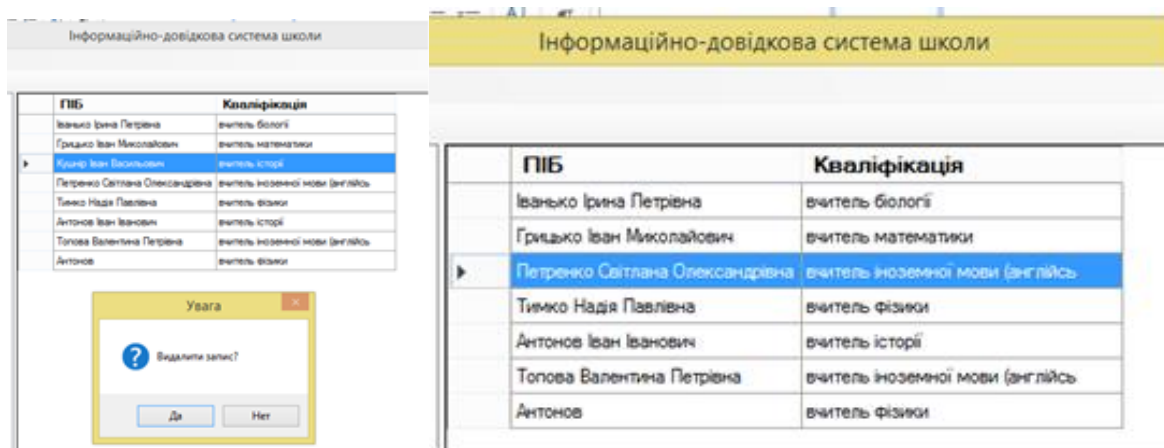


Рис. 3.11. Результати тестування видалення, редагування даних

В результаті підтвердження запис видаляється з бази даних. Аналогічно протестовано видалення і редагування записів в формі Учні, Предмети, Факультативи.

*Висновок:* операції видалення і редагування даних в програмному проєкті відповідають вимогам.

4. Ведення довідників, які будуть використовуватися базою даних. Як довідників виступають таблиці: кваліфікація вчителя, освіта, національність учня, батьки, стан здоров'я.

Завдання: ввести дані в довідник кваліфікація вчителя. Виберемо в меню Довідники: вчителя- кваліфікація. При натисканні кнопки додавання запису внизу списку з'являється порожній запис для введення даних (рис. 3. 12).

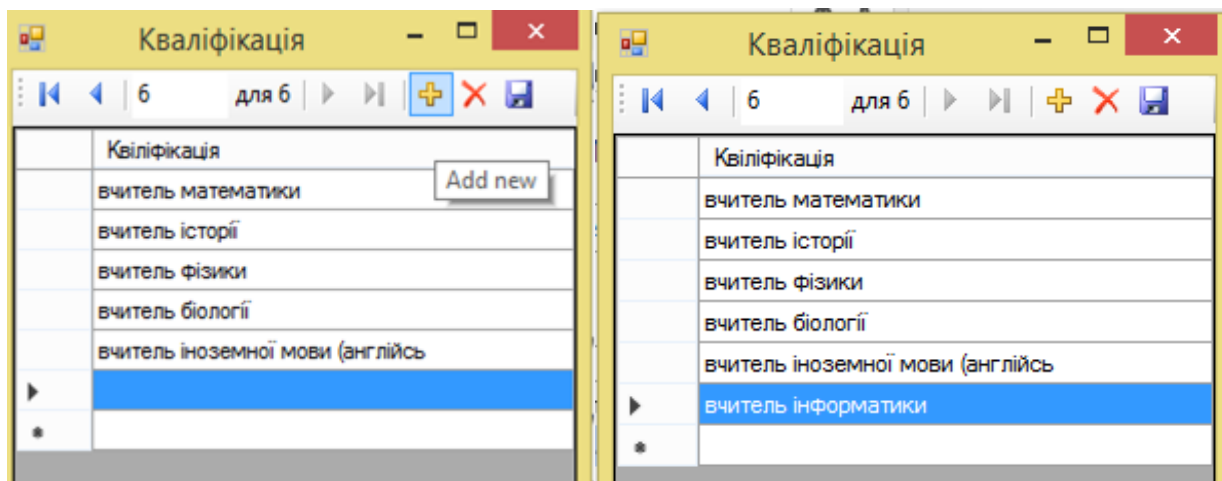


Рис. 3.12. Результати тестування ведення довідників

Аналогічно протестовано додавання даних в інші довідники.

*Висновок:* операції додавання даних в довідники в проєкті Програми відповідають вимогам.

5. Формування звітних форм. Завдання: сформувати звіти: склад учнів за списком, картка навантаження вчителя.

В ході тестування за допомогою меню Звіти були обрані зазначені форми. Приклад звіту відомості успішності учнів за семестр показаний на рис.3.13.

Список учнів 21 квітня 2024р. 16:00

ПІБ	Дата народження	Клас
Іванов Ігор Петрович	15.04.2010 0:00:00	8 А
Петренко Іван Олександрович	21.04.2010 0:00:00	8 А
Онопченко Ірина Петрівна	15.03.2010 0:00:00	8 А

Рис. 3.13. Результати тестування звітів

*Висновок:* операції формування звітних форм в проєкті Програми відповідають вимогам.

## **Висновки до розділу**

Під час розробки клієнтського додатку інформаційно-довідкової системи навчального закладу було проаналізовано можливості технології ADO.NET, що має набір об'єктів, на основі яких можна створювати додатки будь-якого масштабу.

Встановлено, що найбільш оптимізованою мовою для роботи з цією технологією є C #, тому для розробки додатку було обрано середовище Visual Studio. За допомогою ADO.NET Data Provider в додатку здійснено підключення до бази даних FireBird.

Згідно встановлених етапів технології проєктування інформаційних систем було розроблено інформаційно-довідкову систему навчального закладу: "Дніпровська загальноосвітня школа I - III ступенів №5 " на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням.

## ВИСНОВКИ

Під час виконання дипломної роботи було проведено аналіз технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД. Під час дослідження отримані наступні висновки:

- досліджено теоретичні основи інформаційно-довідкових систем: розглянуто поняття ІДС, класифікація, структуру та складові підсистеми;
- проаналізовано засоби проєктування та етапи створення інформаційно-довідкових систем. Встановлено, що технологія розробки ІДС складається з наступних етапів: передпроєктний, проєктний (етапи технічного та робочого проєктування), впровадження, супровід і аналіз функціонування;
- визначені завдання, вимоги та обґрунтування розробки та розроблено технічне завдання на розробку ІДС;
- розглянуто теоретичні підходи до технології клієнт-серверної архітектури, проаналізовано сучасні клієнт-серверні СУБД, які вільно розповсюджуються. Для розробки ІДС обрано FireBird;
- під час методологічного проєктування розроблено інфологічну та даталогічну модель бази даних системи та розроблено фізичну модель бази даних school.gdb;
- під час розробки клієнтського додатку ІДС були обґрунтовано застосовані технологія ADO.NET, мова програмування C #, середовище розробки Visual Studio;
- розроблено інформаційно-довідкову систему навчального закладу: "Дніпровська загальноосвітня школа І - ІІІ ступенів №5" на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. – Київ. – 2017. – 110 с.
2. Антоненко В. М. Сучасні інформаційні системи і технології: управління знаннями : навч. посібник / В. М. Антоненко, С. Д. Мамченко, Ю. В. Рогущина. – Ірпінь : Нац. університет ДПС України, 2016. – 212 с.
3. Воронін А. М. Інформаційні системи прийняття рішень: навчальний посібник. / Воронін А. М., Зіатдінов Ю. К., Климова А. С. – К. : НАУ-друк, 2009. – 136с.
4. Галузинський Г. П. Інформаційні системи у бізнесі. Практикум для індивідуальної роботи: навч.- метод. посіб. для самост. вивч. Дисципліни. / Галузинський Г. П., Денісова О. О., Писаревська Т. А. – К. : КНЕУ, 2008. – 524с.
5. Годун В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М. Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.
6. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортні технології» / О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2010. – 222 с.
7. Інформаційні системи в економіці : навч. посібник / Пономаренко В. С., Золотарьова І. О., Бутова Р. К. та ін. – Х. : Вид. ХНЕУ, 2011. – 176 с.
8. Інформаційні системи в промисловості : навчальний посібник / Л. О. Добровольська, О. О. Черевко. – Маріуполь : ПДТУ, 2014. – 238 с.
9. Інформаційні системи в сучасному бізнесі : навчальний посібник / В. С. Пономаренко, І. О. Золотарьова, Р. К. Бутова та ін. – Х. : Вид. ХНЕУ, 2011. – 484 с.
10. Калінеску Т.В. Інформаційні системи і технології в оподаткуванні: навч. посіб. / Т.В. Калінеску, Г.С. Ліхоносова, О.М. Антіпов. – Луганськ: вид-во СЛУ ім. В. Даля, 2011. – 407 с.
11. Клімушин П. С. Інформаційні системи та технології в економіці : навч.

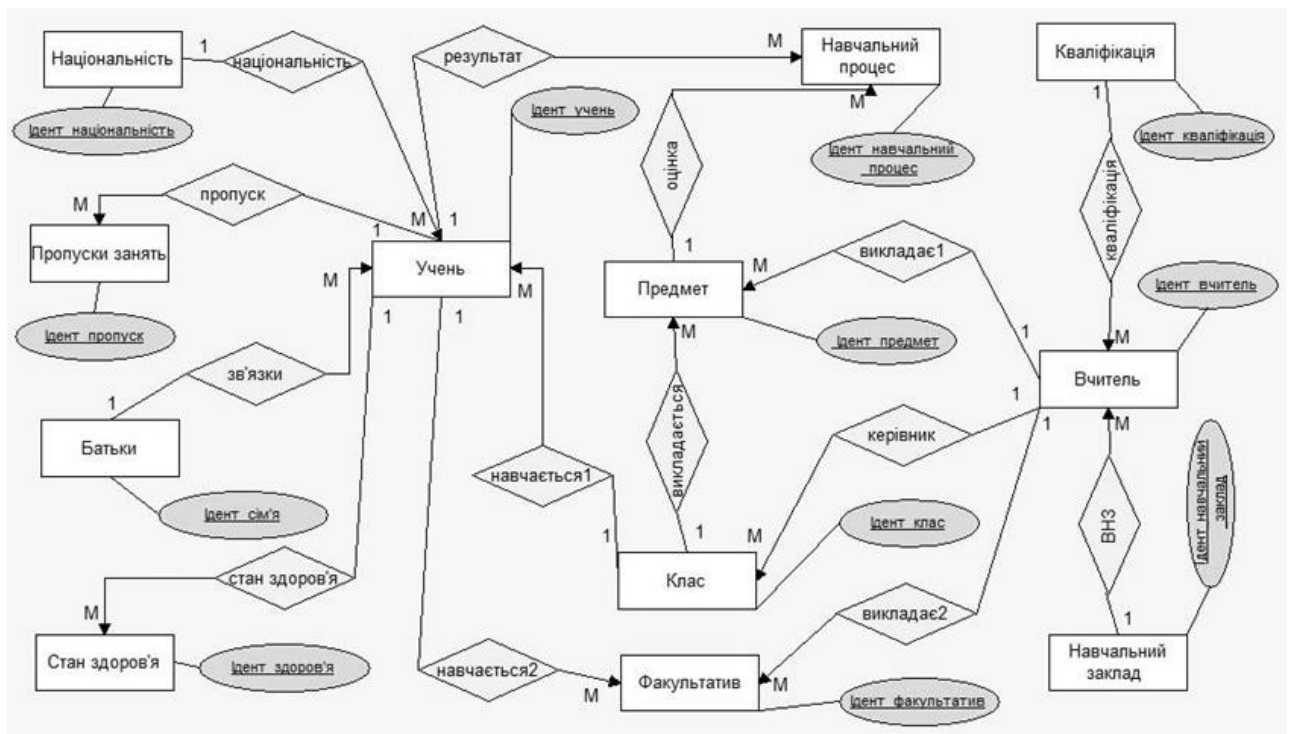
- посіб. / П. С.Клімушин, О.В. Орлов, А.О. Серенок. — Х. : Вид-во ХарPI НАДУ «Магістр», 2011. — 448 с.
- 12.Морзе Н.В. Інформаційні системи. Навч. посібн. /за наук. ред. Н. В. Морзе; Морзе Н.В., Піх О.З. — Івано-Франківськ, «ЛілеяНВ», — 2015. — 384 с.
- 13.Павлиш В. А. Основи інформаційних технологій і систем: Навчальний посібник. / Павлиш В. А., Гліненко Л. К. - Львів: Видавництво Львівської політехніки, 2013. — 500 с.
- 14.Пістунов І. М. Інформаційні системи в фінансово-кредитних установах [текст] навчальний посібник / І. М. Пістунов, Т. В. Борщ. — К.: «Центр учбової літератури», 2013. — 234 с.
- 15.Сендзюк М.А. Інформаційні системи і технології в економіці: навч.-метод. посіб. для самост. вивч. дисципліни / М.А. Сендзюк; М-во освіти і науки України, ДВНЗ “Київ. нац. екон. ун-т ім. В. Гетьмана”. — К. : КНЕУ, 2010. — 68 с.
- 16.Сікірда Ю. В. Інформаційні системи і технології в управлінні зовнішньоекономічною діяльністю : конспект лекцій / Ю. В. Сікірда, А. В. Залевський. — Кіровоград : Видавництво КЛА НАУ, 2013. — 177 с.
- 17.Соколов В.Ю. Інформаційні системи і технології : Навч. посіб. / Соколов В.Ю. — К. : ДУІКТ, 2010. — 138 с.
- 18.Федотова Е.Л. Информационные технологии и системы: учеб. пособие / Е.Л. Федотова. — М.: ИД “ФОРУМ”: ИНФРА-М, 2014. — 352 с.
- 19.Шило С. Г. Інформаційні системи та технології : навчальний посібник / С. Г. Шило, Г. В. Щербак, К. В. Огурцова. — Х. : Вид. ХНЕУ, 2013. — 220 с.
- 20.Юринець В. Є. Інформаційні системи управління персоналом, діловодства і документообігу: навч. посіб. / Юринець В. Є., Юринець Р. В. — Л. : Тріада плюс, 2008. — 628 с.
- 21.Пасічник В.В., Резніченко В. А. Організація баз даних та знань.—К. : Видавнича група «ВНВ», 2006. —384 с.
- 22.Бази даних в інформаційних системах : підруч. / В. І. Гайдаржи, І. В. Ізварін. - К. : Ун-т Україна, 2018. - 418 с.
- 23.Шпеник Т.Б. Організація баз даних. Логічне проектування та робота з

віддаленими базами даних. Методичні вказівки і завдання до лабораторних робіт для студентів 2-го курсу інженерно-технічного факультету спеціальності 123 – «Комп'ютерна інженерія». – Ужгород: «АУТДОРШАРК», 2021. – 79 с.

- 24.Балик Н.Р., Мандзюк В.І. Бази даних MySQL: Навчальний посібник. — Тернопіль: «Навчальна книга – Богдан», 2010.— 160 с.
- 25.Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 1. Організація баз даних та знань. – «Комп'ютинг», 2006. – 460с. 7. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Організація баз даних та знань. – «Комп'ютинг», 2006. – 590с.

## ДОДАТОК А

## Інфологічна модель бази даних інформаційно-довідкової системи



## ДОДАТОК Б

### Програмний код файлу **PlymorphicWidget**

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Data;
using System.Windows.Forms;
namespace UIClient

{
    public enum TableItems { School = 0, Classes = 1, Pupils = 2, Subject = 3, Progress = 4,
Absence = 5, Teachers = 6, Elective = 7 }
    public class PlymorphicWidget
    {
        protected TreeNode nodeData; protected MainWindow widget;
        public PlymorphicWidget(MainWindow window)
        {

            widget = window;
        }
        protected virtual void view(){ } public virtual void adaptFilterA()
        {

            widget.mainGrid.DataSource = null;

        }
        public virtual void adaptFilterB()

        {

            widget.mainGrid.DataSource = null;

        }
        public virtual void adapt()

        {

            widget.mainGrid.DataSource = null;

        }
        public virtual void showEditDialog()

        {
        }

        public virtual void insertDialog()

        {
        }

        public void deleteRow()
        {
```

```

if (widget.mainGrid.DataSource != null)
{
    if (DialogResult.Yes == MessageBox.Show("Видалити запис?", "Увага",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question))
    {
        DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null)
        row.Delete(); else
        MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    else
        MessageBox.Show("Таблиця не визначена!", "Увага", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
}

public virtual void dataSave()

{
}

public virtual void dataRefresh()

{
}

public TreeNode Node

{

get { return nodeData; } set { nodeData = value; }
}

}

public class PupilsTable : PolymorphicWidget

{

public PupilsTable(MainWindow window): base(window) { }
protected override void view()
{
    widget.initViewRelation("relation1", "Національність", 1, 15);
    widget.initViewRelation("relation2", "Клас", 1, 16);
    widget.initViewRelation("relation3", "Батьки", 1, 17);
    widget.mainGrid.Columns[0].Visible = false; widget.mainGrid.Columns[1].Visible = false;
    widget.mainGrid.Columns[2].HeaderText = "ПІБ"; widget.mainGrid.Columns[2].Width = 230;
    widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[4].HeaderText = "Дата
    народження"; widget.mainGrid.Columns[4].Width = 230; widget.mainGrid.Columns[5].Visible =
    false; widget.mainGrid.Columns[6].Visible = false; widget.mainGrid.Columns[7].Visible = false;
    widget.mainGrid.Columns[8].Visible = false; widget.mainGrid.Columns[9].Visible = false;
    widget.mainGrid.Columns[10].Visible = false; widget.mainGrid.Columns[11].Visible = false;
    widget.mainGrid.Columns[12].Visible = false; widget.mainGrid.Columns[13].Visible = false;
    widget.mainGrid.Columns[13].Visible = false; widget.mainGrid.Columns[14].Visible = false;
    widget.mainGrid.Columns[15].Visible = false; widget.mainGrid.Columns[16].Visible = false;
    widget.mainGrid.Columns[17].Visible = false;
}
}

```

```

        public override void adaptFilterA()
        {
            widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject);
            if (widget.pupilsParamDlg.ShowDialog() == DialogResult.OK)
            {
                string filterParemetr = "P_NAME = " + "" + widget.pupilsParamDlg.cbPupilsName.Text +
                "";
                DataView view = new DataView(widget.fbObject.dataTable("PUPILS"), filterParemetr,
                "P_ID_PUPIL", DataViewRowState.CurrentRows);
                if (view.Count == 0)

                    MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
                    MessageBoxIcon.Warning); else
                {
                    widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName;
                    widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                    widget.mainGrid.DataSource = view; this.view();
                }
            }
        }
        public override void adaptFilterB()
        {
            widget.pupilsBirthdayParam = new PupilsBirthdayParam();

            if (widget.pupilsBirthdayParam.ShowDialog() == DialogResult.OK)

            {
                string filterParemetr = "P_BIRTH_DATE >= " + "" +
                widget.pupilsBirthdayParam.date1.Value.ToShortDateString() + "" + " AND " +
                "P_BIRTH_DATE <= " + "" +

                widget.pupilsBirthdayParam.date2.Value.ToShortDateString() + "";
                DataView view = new
                DataView(widget.fbObject.dataTable("PUPILS"), filterParemetr, "P_ID_PUPIL",
                DataViewRowState.CurrentRows);
                if (view.Count == 0)

                    MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
                    MessageBoxIcon.Warning); else
                {
                    widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName;
                    widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                    widget.mainGrid.DataSource = view; this.view();
                }
            }
        }
        public override void adapt()
        {
            widget.mainGrid.DataSource = 0;
            DataView view = widget.pupilsView[widget.treePosition[nodeData.Text]];
            widget.currentTable = view.Table.TableName;
            widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
            widget.mainGrid.DataSource = view;
            this.view();
        }
    }
}

```

```

    }
    public override void showEditDialog()
    {
        widget.pplDlg = new PupilsDialog(widget.fbObject); DataRow row =
        widget.findCurrentRow(widget.mainGrid); if (row != null)
        widget.pplDlg.ShowDialog(row); else
        MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }

    public override void insertDialog()
    {
        widget.pplDlg = new PupilsDialog(widget.fbObject); widget.pplDlg.ShowDialog();
    }

    public override void dataSave()
    {
        widget.fbObject.save("PUPILS");
    }

    public override void dataRefresh()
    {
        widget.fbObject.refill("PUPILS");
    }
}

public class SubjectTable : PolymorphicWidget
{
    public SubjectTable(MainWindow window) : base(window) { }
    protected override void view()
    {
        widget.initViewRelation("relation11", "Вчитель", 1, 0);

        widget.initViewRelation("relation14", "Клас", 1, 1);
        widget.mainGrid.Columns[2].Visible = false;
        widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[4].Visible = false;
        widget.mainGrid.Columns[5].HeaderText = "Назва предмета";
        widget.mainGrid.Columns[5].Width = 230; widget.mainGrid.Columns[6].HeaderText = "Кількість
        годин"; widget.mainGrid.Columns[6].Width = 230;
    }
}

```

```

public override void adaptFilterA()

{

    widget.teacherParamDlg = new TeacherNameParam(widget.fbObject);    if
(widget.teacherParamDlg.ShowDialog() == DialogResult.OK)
    {

        string filterParemetr1 = "T_NAME = " + "" + widget.teacherParamDlg.cbTeacher.Text + "";
        DataView view1 = new
            DataView(widget.fbObject.dataTable("TEACHERS"), filterParemetr1, "T_ID_TEACHER",
            DataViewRowState.CurrentRows);
        if (view1.Count == 0)

            MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
            MessageBoxIcon.Warning); else
            {

                string filterParemetr2 = "S_ID_TEACHER = " + "" + view1[0].Row[0].ToString() + "";
                DataView view2 = new DataView(widget.fbObject.dataTable("SUBJECT"), filterParemetr2,
                "S_ID_SUBJECT", DataViewRowState.CurrentRows);
                if (view2.Count == 0)

                    {
                        MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
                        MessageBoxIcon.Warning);
                        return;
                    }

                widget.mainGrid.DataSource = 0; widget.currentTable = view2.Table.TableName;
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = view2; this.view();
            }

        }

    }

}

public override void adaptFilterB()

{

}

public override void adapt()

{

    widget.mainGrid.DataSource = 0; DataView view =
        widget.subjectView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =
view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view;
        this.view();
    }
}

```

```

public override void showEditDialog()
{

    widget.sbjDlg = new SubjectDialog(widget.fbObject);    DataRow row =
widget.findCurrentRow(widget.mainGrid); if (row != null)
    widget.sbjDlg.ShowDialog(row); else
    MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

public override void insertDialog()

{

widget.sbjDlg = new SubjectDialog(widget.fbObject); widget.sbjDlg.ShowDialog();
}

public override void dataSave()

{

widget.fbObject.save("SUBJECT");

}

public override void dataRefresh()

{

widget.fbObject.refill("SUBJECT");

}

}

public class ProgressTable : PolymorphicWidget

{

public ProgressTable(MainWindow window) : base(window) { }
protected override void view()

{
    widget.initViewRelation("relation13", "Учень", 3, 0);
    widget.initViewRelation("relation5", "Предмет", 2, 0);
    widget.mainGrid.Columns[2].Visible = false; widget.mainGrid.Columns[3].Visible = false;
widget.mainGrid.Columns[4].Visible = false; widget.mainGrid.Columns[8].Visible = false;
    widget.mainGrid.Columns[5].HeaderText = "1-й семестр";
widget.mainGrid.Columns[5].Width = 140; widget.mainGrid.Columns[6].HeaderText = "2-й
семестр"; widget.mainGrid.Columns[6].Width = 140; widget.mainGrid.Columns[7].HeaderText =
"Оцінка за рік"; widget.mainGrid.Columns[7].Width = 140;
}
public override void adaptFilterA()

{

```

```

        widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if
(widget.pupilsParamDlg.ShowDialog() == DialogResult.OK)
    {
        string filterParemetr1 = "P_NAME = " + "" + widget.pupilsParamDlg.cbPupilsName.Text +
""; DataView view1 = new
        DataView(widget.fbObject.dataTable("PUPILS"), filterParemetr1, "P_ID_PUPIL",
DataViewRowState.CurrentRows);
        if (view1.Count == 0)

            MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
MessageBoxIcon.Warning); else
            {
                string filterParemetr2 = "P_ID_PUPIL = " + "" + view1[0].Row[0].ToString() + "";
                DataView view2 = new DataView(widget.fbObject.dataTable("PROGRESS"),
filterParemetr2, "P_ID_PROGRESS", DataViewRowState.CurrentRows);
                if (view2.Count == 0)

                    {

                        MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
MessageBoxIcon.Warning); return;
                    }

                widget.mainGrid.DataSource = 0; widget.currentTable = view2.Table.TableName;
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = view2; this.view();
            }
        }

        public override void adaptFilterB()

        {
        }

        public override void adapt()

        {

            widget.mainGrid.DataSource = 0; DataView view =
            widget.progresView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =
            view.Table.TableName;
            widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
            widget.mainGrid.DataSource = view; this.view();
        }
        public override void showEditDialog()
        {
            widget.prgDlg = new ProgressDialog(widget.fbObject); DataRow row =
            widget.findCurrentRow(widget.mainGrid); if (row != null)
                widget.prgDlg.ShowDialog(row); else
                MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
}

```

```

    }
    public override void insertDialog()
    {

        widget.prgDlg = new ProgressDialog(widget.fbObject); widget.prgDlg.ShowDialog();
    }

    public override void dataSave()

    {

        widget.fbObject.save("PROGRESS");

    }

    public override void dataRefresh()

    {

        widget.fbObject.refill("PROGRESS");

    }

    }
    public class AbsenceTable : PolymorphicWidget

    {

        public AbsenceTable(MainWindow window) : base(window) { }
        protected override void view()

        {

            widget.initViewRelation("relation16", "Клас", 1, 0);
            widget.initViewRelation("relation6", "Учень", 2, 0);
            widget.mainGrid.Columns[1].Visible = false; widget.mainGrid.Columns[2].Visible = false;
            widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[6].Visible = false;

            widget.mainGrid.Columns[4].HeaderText = "Місяць"; widget.mainGrid.Columns[4].Width =
140; widget.mainGrid.Columns[5].HeaderText = "Кількість годин";
            widget.mainGrid.Columns[5].Width = 170;
        }

        public override void adaptFilterA()

        {

            widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if
(widget.pupilsParamDlg.ShowDialog() == DialogResult.OK)
            {

                string filterParemetr1 = "P_NAME = " + "" + widget.pupilsParamDlg.cbPupilsName.Text +
""; DataView view1 = new

```

```

        DataView(widget.fbObject.dataTable("PUPILS"), filterParemetr1, "P_ID_PUPIL",
DataViewRowState.CurrentRows);
        if (view1.Count == 0)
            MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        else

            {

                string filterParemetr2 = "A_ID_PUPIL = " + "" + view1[0].Row[0].ToString() + "";
                DataView view2 = new DataView(widget.fbObject.dataTable("ABSENCE"), filterParemetr2,
"A_ID_ABSENCE", DataViewRowState.CurrentRows);
                if (view2.Count == 0)

                    {

                        MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
MessageBoxIcon.Warning); return;
                    }

                widget.mainGrid.DataSource = 0; widget.currentTable = view2.Table.TableName;
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = view2; this.view();
            }

        }

        public override void adaptFilterB()

        {
        }

        public override void adapt()
        {
            widget.mainGrid.DataSource = 0; DataView view =
            widget.absenceView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =
view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view;
            this.view();
        }

        public override void showEditDialog()

        {

            widget.absDlg = new AbsenceDialog(widget.fbObject); DataRow row =
            widget.findCurrentRow(widget.mainGrid); if (row != null)
                widget.absDlg.ShowDialog(row); else
                MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }

        public override void insertDialog()

        {

            widget.absDlg = new AbsenceDialog(widget.fbObject); widget.absDlg.ShowDialog();
        }

```

```

public override void dataSave()

{
    widget.fbObject.save("ABSENCE");
}
public override void dataRefresh()
{
    widget.fbObject.refill("ABSENCE");

}

}

public class TeacherTable : PolymorphicWidget
{
    public TeacherTable(MainWindow window) : base(window) { }
    protected override void view()
    {
        widget.initViewRelation("relation8", "Кваліфікація", 1, 17);

        widget.initViewRelation("relation9", "Освіта", 1, 18);
        widget.mainGrid.Columns[0].Visible = false;
        //widget.mainGrid.Columns[1].Visible = false; widget.mainGrid.Columns[2].Visible = false;
        widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[4].Visible = false;
        widget.mainGrid.Columns[5].Visible = false; widget.mainGrid.Columns[6].Visible = false;
        widget.mainGrid.Columns[7].Visible = false; widget.mainGrid.Columns[8].Visible = false;
        widget.mainGrid.Columns[9].Visible = false; widget.mainGrid.Columns[10].Visible = false;
        widget.mainGrid.Columns[11].Visible = false; widget.mainGrid.Columns[12].Visible = false;
        widget.mainGrid.Columns[13].Visible = false; widget.mainGrid.Columns[14].Visible = false;
        widget.mainGrid.Columns[15].Visible = false; widget.mainGrid.Columns[16].Visible = false;
        widget.mainGrid.Columns[18].Visible = false;
        widget.mainGrid.Columns[1].HeaderText = "ІІБ"; widget.mainGrid.Columns[14].Width =
230;
    }
    public override void adaptFilterA()

    {

        widget.teacherParamDlg = new TeacherNameParam(widget.fbObject); if
(widget.teacherParamDlg.ShowDialog() == DialogResult.OK)
        {

            string filterParemetr = "T_NAME = " + "" + widget.teacherParamDlg.cbTeacher.Text + "";
            DataView view = new
            DataView(widget.fbObject.dataTable("TEACHERS"), filterParemetr, "T_ID_TEACHER",
            DataRowState.CurrentRows);
            if (view.Count == 0)

                MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
            else
            {
                widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName;
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = view; this.view();
            }
        }
    }
}

```

```

    }
    }

    }
    public override void adaptFilterB()
    {

        widget.teacherCategoryParamDlg = new TeacherCategoryParam();
        if (widget.teacherCategoryParamDlg.ShowDialog() == DialogResult.OK)

        {
            string filterParemetr = "T_CATEGORY = " + "" +
            widget.teacherCategoryParamDlg.cbTextParam.Text + "";
            DataView view = new DataView(widget.fbObject.dataTable("TEACHERS"), filterParemetr,
            "T_ID_TEACHER", DataViewRowState.CurrentRows);
            if (view.Count == 0)

                MessageBox.Show("Дані відсутні в базі", "Увага!", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
            else

            {

                widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName;
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = view; this.view();
            }
            }
            }
            public override void adapt()
            {
                widget.mainGrid.DataSource = 0; widget.currentTable = "TEACHERS";
                widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
                widget.mainGrid.DataSource = widget.bindingSource1; this.view();
            }
            public override void showEditDialog()
            {
                widget.tchDlg = new TeachersDialog(widget.fbObject); DataRow row =
                widget.findCurrentRow(widget.mainGrid); if (row != null)
                widget.tchDlg.ShowDialog(row); else
                MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            }

            public override void insertDialog()
            {
                widget.tchDlg = new TeachersDialog(widget.fbObject); widget.tchDlg.ShowDialog();
            }
            public override void dataSave()
            {
                widget.fbObject.save("TEACHERS");
            }
            public override void dataRefresh()

```

```

    {
        widget.fbObject.refill("TEACHERS");
    }
}

public class ElectiveTable : PolymorphicWidget
{
    public ElectiveTable(MainWindow window) : base(window) { }
    protected override void view()
    {
        widget.initViewRelation("relation7", "Учень", 2, 4);
        widget.initViewRelation("relation12", "Вчитель", 1, 5);
        widget.mainGrid.Columns[0].Visible = false; widget.mainGrid.Columns[2].Visible = false;
        widget.mainGrid.Columns[3].Visible = false;
        widget.mainGrid.Columns[1].HeaderText = "Факультатив";
        widget.mainGrid.Columns[1].Width = 230; widget.mainGrid.Columns[6].HeaderText = "Кількість
        годин"; widget.mainGrid.Columns[6].Width = 230;
    }
    public override void adaptFilterA()
    {
    }
    public override void adaptFilterB()
    {
    }
    public override void adapt()
    {
        widget.mainGrid.DataSource = 0;
        widget.currentTable = "ELECTIVE"; widget.bindingSource1.DataSource =
        widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource =
        widget.bindingSource1; this.view();
    }
    public override void showEditDialog()
    {
        widget.elcDlg = new ElectiveDialog(widget.fbObject); DataRow row =
        widget.findCurrentRow(widget.mainGrid); if (row != null)
        widget.elcDlg.ShowDialog(row); else
        MessageBox.Show("Рядок не знайдено", "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    public override void insertDialog()
    {
        widget.elcDlg = new ElectiveDialog(widget.fbObject); widget.elcDlg.ShowDialog();
    }

    public override void dataSave()
    {
        widget.fbObject.save("ELECTIVE");
    }
    public override void dataRefresh()
    {
        widget.fbObject.refill("ELECTIVE");
    }
}

public class WidgetAgregat

```

```

{
private PolymorphicWidget[] polymorphWidget; private const int size = 7;
public WidgetAgregat(MainWindow window)
{
    polymorphWidget      = new    PolymorphicWidget[size]; polymorphWidget[0] = new
PolymorphicWidget(window); polymorphWidget[1] = new PupilsTable(window); polymorphWidget[2]
= new    SubjectTable(window);  polymorphWidget[3]  = new    ProgressTable(window);
polymorphWidget[4]      = new    AbsenceTable(window);  polymorphWidget[5]      = new
TeacherTable(window); polymorphWidget[6] = new ElectiveTable(window);
    }
    public PolymorphicWidget this[int index]
    {
        get

        {
            if (index < 0 || index >= size) return polymorphWidget[0];
            return polymorphWidget[index];
        }
    }
}

```

## ДОДАТОК В

### Програмний код файлу FirebirdInterface

```
using System;
using System.Collections;
using System.Collections.Generic; using System.Data;
using System.Text; using System.Linq;
using System.Windows.Forms;
using FirebirdSql.Data.FirebirdClient; using System.Text.RegularExpressions;
namespace UIClient

{

    public class FirebirdInterface
    {
        private List<FbDataAdapter> adapters; private FbConnection connect;
        private DataSet memory_db;
        public FbConnection ConnectionObject
        {

            get { return connect; }

        }
        public FirebirdInterface()
        {
            connect      = new FbConnection(); adapters = new List<FbDataAdapter>(); memory_db =
new DataSet();
        }
        private string queryGetTables()
        {
            return "SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE
(RDB$SYSTEM_FLAG = 0) AND (RDB$VIEW_SOURCE IS NULL) ORDER BY
RDB$RELATION_NAME";
        }
        private string queryGetFields(string tableName)
        {
            return "SELECT R.RDB$FIELD_NAME, R.RDB$NULL_FLAG, T.RDB$TYPE_NAME,
F.RDB$FIELD_LENGTH, F.RDB$FIELD_SCALE, F.RDB$FIELD_SUB_TYPE " +
"FROM RDB$TYPES T, RDB$RELATION_FIELDS R " + "INNER JOIN RDB$FIELDS
F ON F.RDB$FIELD_TYPE =
T.RDB$TYPE AND T.RDB$FIELD_NAME = 'RDB$FIELD_TYPE' " + "WHERE
F.RDB$FIELD_NAME = R.RDB$FIELD_SOURCE AND
R.RDB$SYSTEM_FLAG = 0 AND RDB$RELATION_NAME = '" + tableName
+
"" ORDER BY R.RDB$FIELD_POSITION ASC";
        }

        private DataColumn createDataColumn(string columnName, string typeName, int subType,
bool allowDBNull, int lenght)
        {
            DataColumn column = new DataColumn(columnName); column.AllowDBNull =
allowDBNull;
            switch (typeName)
            {
```

```

        case ("SHORT")           : { column.DataType = typeof(Int16); break; } case ("LONG")
                                   : { column.DataType = typeof(Int32); break; }
        case ("INT64")           : { column.DataType = subType == 0 ? typeof(Int64)
                                   : typeof(Decimal); break; }
        case ("FLOAT")           : { column.DataType = typeof(Single); break;
        }
        case ("DOUBLE")          : { column.DataType = typeof(Double); break; }
        case ("DATE")             : { column.DataType = typeof(DateTime);
column.DateTimeMode = DataSetDateTime.Local; break; }
        case ("TIMESTAMP"): { column.DataType = typeof(DateTime); column.DateTimeMode =
DataSetDateTime.Local; break; }
        case ("TIME")             : { column.DataType = typeof(TimeSpan); break; } case ("TEXT")
                                   : { column.DataType = typeof(String);
        column.MaxLength = lenght; break; }
        case ("VARYING") : { column.DataType = typeof(String); column.MaxLength = lenght;
break; }
        case ("CSTRING") : { column.DataType = typeof(String); column.MaxLength = lenght;
break; }
        case ("BLOB")            : { column.DataType = typeof(byte[]); column.MaxLength =
lenght; break; }
        }
        return column;
    }
    private void loadTableColumn(string tableName)
    {
        int index = memory_db.Tables.IndexOf(tableName); if (index >= 0)
        {
            if (!isOpen()) openDataBase();
            DataTable table = new DataTable("FIELDSNAME"); FbDataAdapter adapter = new
FbDataAdapter(queryGetFields(tableName), connect); adapter.Fill(table);
            int tableCount = table.Rows.Count; for (int i = 0; i < tableCount; ++i)
            {
                string columnName = table.Rows[i][0].ToString().TrimEnd();
                bool allowDBNull = table.Rows[i][1] != DBNull.Value ? false : true; string typeName =
table.Rows[i][2].ToString().TrimEnd();
                int lenght = table.Rows[i][3] != DBNull.Value ? Convert.ToInt32(table.Rows[i][3]) : 0;
                int subTypeIndex = table.Rows[i][5] != DBNull.Value ? Convert.ToInt32(table.Rows[i][5]) :
0;
                DataColumn column = createDataColumn(columnName, typeName, subTypeIndex,
allowDBNull, lenght);
                memory_db.Tables[index].Columns.Add(column);
            }
            memory_db.Tables[tableName].PrimaryKey = new DataColumn[] {
memory_db.Tables[tableName].Columns[0] };
            memory_db.Tables[tableName].Columns[0].AutoIncrement = true;
            fillSchema(tableName);
            if (isOpen()) closeDataBase();
        }
    }
    public string initConnectionString(string clientLibrary, string database, string userID, string
password, string role, string charset, int port)
    {
        FbConnectionStringBuilder connectionString = new FbConnectionStringBuilder();
        connectionString.ClientLibrary = clientLibrary; connectionString.Database = database;

```

```

connectString.UserID = userID;
    connectString.Password = password; connectString.Charset = charset; connectString.Role =
role; connectString.Port = port;
    connect.ConnectionString = connectString.ConnectionString; return
connect.ConnectionString;
    }
    public bool openDataBase()
    {
    try
    {
    if (connect.ConnectionString.Length > 0) connect.Open();
    else
    throw new System.Ехception("Рядок з'єднання з БД не коректний");
    }
    catch (System.Ехception Except)
    {
    MessageBox.Show(Except.Message, "Помилка", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
    connect.Close(); return false;
    }

    return connect.State == System.Data.ConnectionState.Open ? true : false;
    }
    public bool closeDataBase()
    {
    try { connect.Close();
    }
    catch (System.Ехception Except)
    {
    MessageBox.Show(Except.Message, "Помилка", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
    return false;
    false;
    }
    }
    return connect.State == System.Data.ConnectionState.Closed ? true :
    public bool isOpen()
    {
    return connect.State == System.Data.ConnectionState.Open ? true : false;
    }
    public List<string> listTables()
    {
    List<string> list = new List<string>();
    for (int i = 0; i < memory_db.Tables.Count; ++i) list.Add(memory_db.Tables[i].TableName);
    return list;
    }
    public List<string> listColumns(int indexTable)
    {
    List<string> list = new List<string>();
    if (tableName(indexTable) != string.Empty)

    {
    int count = memory_db.Tables[indexTable].Columns.Count; for (int i = 0; i < count; ++i)
    list.Add(memory_db.Tables[indexTable].Columns[i].ColumnName);
    }
    }

```

```

    }
    return list;
    }
    public void loadTables()
    {
        if (!isOpen()) openDataBase(); this.clear();
        connect();
        Regex rgx = new Regex(@"\$");

        DataTable table = new DataTable("TABLESNAME"); FbDataAdapter adapter = new
FbDataAdapter(queryGetTables(), adapter.Fill(table);
        int tableCount = table.Rows.Count;
        for (int i = 0, j = 0; i < tableCount; ++i)
        {
            string item = table.Rows[i][j].ToString(); item = item.TrimEnd();
            if (!rgx.IsMatch(item))
            {
                adapters.Add(new FbDataAdapter()); FbCommandBuilder commandBuilder = new
FbCommandBuilder(adapters.Last());
                commandBuilder.ConflictOption = ConflictOption.OverwriteChanges;
                memory_db.Tables.Add(new DataTable(item));
            }
        }

        if (isOpen()) closeDataBase();

    }
    public void fillSchema(string tableName)
    {
        try
        {
            int index = memory_db.Tables.IndexOf(tableName);

            if (index >= 0 && connect.ConnectionString.Length > 0)

            {

                adapters[index].SelectCommand = new FbCommand("SELECT * FROM " + tableName,
connect);
                adapters[index].FillSchema(memory_db, System.Data.SchemaType.Mapped);
            }

            else

            throw new Exception("Помилка! Не вірне ім'я таблиці або рядок з'єднання");
        }

        catch (Exception e)

        {

            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

```

```

    }
    public int tableIndex(string name)
    {
        try
        {
            int index = memory_db.Tables.IndexOf(name); if (index == -1)
            throw new Exception("Таблиці з цим ім'ям не існує"); return index;
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MemoryBoxIcon.Error);
            return -1;
        }
    }
    public string tableName(int index)
    {
        try
        {
            if (index >= 0 && index <= memory_db.Tables.Count - 1) return
memory_db.Tables[index].TableName;
            else
            throw new Exception("Таблиці з цим індексом не існує");
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MemoryBoxIcon.Error);
            return string.Empty;
        }
    }
    public DataTable dataTable(string tableName)
    {
        int index = tableIndex(tableName); if (index != -1)
        return memory_db.Tables[index]; else
        return null;
    }
    public FbDataAdapter tableAdapter(string tableName)
    {
        int index = tableIndex(tableName); if (index != -1)
        return adapters[index]; else
        return null;
    }
    public DataSet dataSet()
    {
        return memory_db;
    }
    public bool refill(string tableName)
    {
        try
        {
            int index = memory_db.Tables.IndexOf(tableName);

```

```

        if (index >= 0 && index <= memory_db.Tables.Count - 1)
        {
            adapters[index].Fill(memory_db, tableName);
            return true;
        }
        else return false;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        return false;
    }
}

public bool fill(String tableName)
{
    try
    {
        int index = memory_db.Tables.IndexOf(tableName);
        if (index >= 0 && index <= memory_db.Tables.Count - 1)
        {
            if (memory_db.Tables.Contains(tableName)) memory_db.Tables[index].Clear();
            this.loadTableColumn(tableName); adapters[index].Fill(memory_db, tableName); return
true;
        }
        else return false;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        return false;
    }
}

public bool save(string tableName)
{
    DataTable currentTable = dataTable(tableName);
    if (currentTable.GetChanges() == null) return false; try
    {
        tableAdapter(tableName).Update(currentTable.GetChanges());
        currentTable.AcceptChanges();
        return true;
    }
    catch (Exception)
    {
        MessageBox.Show("Не вдалося зберегти запис", "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        currentTable.RejectChanges(); return false;
    }
}

public void clear()
{
    adapters.Clear(); memory_db.Relations.Clear(); memory_db.Tables.Clear();
}

```

```

memory_db.Clear();
    }
    public void clear(int index)
    {
        try
        {
            string item = tableName(index); if (item == string.Empty)
            {
                adapters.Remove(adapters[index]); memory_db.Tables[index].Clear();
            }
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MemoryMessageBoxIcon.Error);
        }
    }
    public void clear(string tableName)

    {
        try
        {
            int index = tableIndex(tableName); if (index != -1)
            {
                adapters.Remove(adapters[index]); memory_db.Tables[index].Clear();
            }
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MemoryMessageBoxIcon.Error);
        }
    }
    public void setRelation(string relationName, string parentTable, string primaryKey, string
childrenTable, string forigenKey)
    {
        try

        {
            memory_db.Relations.Add(new DataRelation(relationName,
dataTable(parentTable).Columns[primaryKey], dataTable(childrenTable).Columns[forigenKey],
true));
        }
        catch (Exception e)

        {
            MessageBox.Show(e.Message, "Помилка", MessageBoxButtons.OK,
MemoryMessageBoxIcon.Error);
        }
    }
    public DataTable executeQuery(string strQuery)

    {
        DataTable table = new DataTable("QUERY");

```

```
FbDataAdapter adapter = new FbDataAdapter(strQuery, connect); adapter.Fill(table);  
return table;  
}  
}
```