Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Розподілена комп'ютерна система збору та аналізу даних щодо захворюваності на COVID-19*

Виконав(ла): студент(ка) *IV* курсу, групи *СІс-44*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

(підпис) *Бєляєв Ю.В.* (прізвище та ініціали)

Керівник (підпис) *Луцків А.М.* (прізвище та ініціали)

Нормоконтроль (підпис) *Луцик Н.С.* (прізвище та ініціали)

Завідувач кафедри (підпис) *Осухівська Г.М.* (прізвище та ініціали)

Рецензент (підпис) *Млинко Б.Б.* (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« » 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Бєляєву Юрію Віталійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розподілена комп'ютерна система збору та аналізу даних щодо захворюваності на COVID-19*

Керівник роботи *Луцків Андрій Мирославович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *10* » *лютого* 2021 року № *4.7-97*

2. Термін подання студентом завершеної роботи *25.06.2021 р.*

3. Вихідні дані до роботи *Інформація про структуру даних, які необхідно зберігати, тип розподіленої комп'ютерної системи, архітектура системи – клієнт-сервер, тип програмного забезпечення – desktop*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання. 2. Проектування розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19. 3. Реалізація програмного забезпечення розподіленої комп'ютерної системи. 4. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Архітектура розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19
2. Інформаційні потоки розподіленої комп'ютерної системи
3. ER-діаграми бази даних
4. Архітектура програмного забезпечення локальних вузлів комп'ютерної системи
5. Алгоритм формування звітної документації
6. Загальний алгоритм роботи програмного забезпечення

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Пилипець М.І., д.т.н., проф. каф. МТ*		
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз вимог до комп'ютерної системи*	*10.02-20.02.2021*	
2.	*Аналіз можливих рішень щодо проектування розподілених комп'ютерних систем*	*20.02-28.02.2021*	
3.	*Проектування розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19*	*28.02-30.03.2021*	
4.	*Реалізація програмного забезпечення розподіленої комп'ютерної системи.*	*30.03-07.05.2021*	
5.	*Безпека життєдіяльності, основи охорони праці.*	*07.05-25.05.2021*	
6.	*Оформлення пояснювальної записки кваліфікаційної роботи*	*25.05-12.06.2021*	
7	*Попередній захист кваліфікаційної роботи*	*12.06-16.06.2021*	
8.	*Захист кваліфікаційної роботи*	*20.06-27.06.2021*	

Студент _____ *Бєляєв Юрій Віталійович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцків Андрій Мирославович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19 // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Бєляєв Юрій Віталійович // ТНТУ, спеціальність 123 «Комп’ютерна інженерія»// Тернопіль, 2021 // с.– 80 , рис. – 30 , табл. – 32, аркушів А1 – 6, бібліогр. – 16.

Ключові слова: комп’ютерна система, збір, аналіз, дані, захворюваність, COVID-19.

Метою даної кваліфікаційної роботи бакалавра є проектування та реалізація розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19.

У кваліфікаційній роботі бакалавра спроектовано архітектуру розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19, розроблено програмне забезпечення для підтримки відповідних бізнес-процесів та формування статистичної звітності за обраний період часу.

У системі реалізовано можливість заповнення довідників щодо осіб, які проходять тестування на COVID-19, регіонів та населених пунктів, підприємств на яких працюють особи та ряду інших. Усі довідники у системі категоризовані, що забезпечує зручність та швидкість пошуку інформації.

Окрім того, на локальних вузлах розподіленої комп’ютерної системи, розгорнуто програмне забезпечення для забезпечення можливості реєстрації пацієнтів, фіксації результатів тестування, формування звітів і статистичних даних в розрізі різних критеріїв: дати і часу, віку осіб і т.п.

ABSTRACT

Distributed computer-aided system of COVID-19 number of cases data collection and analysis // Bachelor’s thesis // Bieliaev Yurii Vitaliyovych// TNTU, speciality 123 «Computer engineering»// Ternopil, 2021 // p.– 80 , fig. – 30 , tab. – 32, posters A1 – 6, ref. – 16.

Keywords: computer system, gathering, analysis, data, illness, COVID-19

The purpose of this bachelor's thesis is to design and implement a distributed computer system for the collection and analysis of data on the incidence of COVID-19.

The bachelor's thesis designed the architecture of a distributed computer system for collecting and analyzing data on the incidence of COVID-19, developed software to support relevant business processes and generate statistical reporting for the selected period of time.

The system provides the ability to fill in directories for people who are tested for COVID-19, regions and settlements, enterprises where people work and a number of others. All directories in the system are categorized, which provides convenience and speed of information retrieval.

In addition, at local nodes of the distributed computer system, software is deployed to provide the ability to register patients, record test results, generate reports and statistics in terms of various criteria: date and time, age, etc.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

БД	База даних
КС	Комп’ютерна система
ПЗ	Програмне забезпечення
ПО	Предметна область
РКС	Розподілена комп’ютерна система
СКБД	Система керування базами даних
ER	Entity-Relationships
UML	Unified Modelling Language

ВСТУП

Сучасні комп’ютерні системи забезпечують автоматизацію бізнес-процесів у різних сферах людської діяльності. Спектр їхнього застосування дуже широкий, починаючи від простих систем обліку та аналізу даних у торгівлі, закінчуючи критичними системами, які мають безпосередній вплив на життя і здоров’я людей.

У теперішніх умовах, спричинених поширенням пандемії COVID-19, актуальними та важливими задачами є збір, фіксація та аналіз даних щодо захворюваності на різних географічно-розподілених територіях.

В Україні збором та аналізом даних щодо захворюваності пандемічним вірусом займаються обласні лабораторні центри, утворені на базі колишніх санітарно-епідеміологічних служб. Процес збору і формування статистики поширення COVID-19 без засобів автоматизації є неефективним та безрезультатним, тому потребує розробки комплексного рішення щодо впровадження відповідної комп’ютерної системи.

Оскільки дані щодо захворюваності певних окремих осіб є конфіденційними і не підлягають розголошенню, тому необхідно забезпечити їх захист і розробити процедури авторизованого доступу до них. Окрім цього, важливим є правильна організація архітектури комп’ютерної системи, яка б забезпечувала ефективність, продуктивність, адекватність і достовірність збору та опрацювання інформації. Це ще раз підкреслює актуальність розробки розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19.

Метою кваліфікаційної роботи для здобуття освітнього ступеня бакалавра є проектування архітектури, алгоритмів та засобів доступу до інформації, що в комплексі формують розподілену комп’ютерну систему щодо захворюваності на COVID-19.

Для досягнення мети роботи пропонується використати підхід побудови розподілених комп’ютерних систем в основі яких лежить реалізація

програмно-апаратних desktop систем на локальному рівні і централізоване керування та передача даних у сховище, яке може бути розміщене у хмарному сервісі.

Для реалізації такої комп’ютерної системи необхідно провести аналіз існуючих рішень, визначити і деталізувати вимоги до розподіленої комп’ютерної системи, обґрунтувати вибір оптимальних засобів проектування та реалізації і як результат – розробити прототип системи. Саме дослідженню і розв’язку таких задач присвячена кваліфікаційна робота на здобуття освітнього ступеня бакалавра.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вимог до комп'ютерної системи

Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19 призначена для автоматизації процесу збору, накопичення, опрацювання і централізованого керування статистичними показниками щодо захворюваності пандемічним вірусом COVID-19. Дана система може бути використана обласними лабораторними центрами, утвореними на базі санітарно-епідеміологічних служб, та повинна забезпечувати щоденний моніторинг стану захворюваності в окремих регіонах. Окрім цього, розподілена комп’ютерна система повинна володіти функціональністю щодо формування та надсилання звітів у централізоване сховище і надавати авторизований доступ передбаченим типам користувачів. Основна функціональність системи збору та аналізу даних передбачає зручне опрацювання статистичної інформації, візуалізацію та побудову відповідних видів діаграм і графіків у відповідності до показників, виводити кількість нових випадків захворюваності за останній звітний період установ, забезпечувати централізоване зберігання даних, модифікацію інформації користувачами в межах їх компетентності.

Метою створення системи є автоматизація процесу збору та опрацювання даних щодо поширення захворюваності на COVID-19 на основі реалізації розподіленої комп’ютерної системи.

До основних задач, які необхідно розв’язати для досягнення мети, входять:

– розробка ефективного механізму розподіленого зберігання та моніторингу статистичних показників;

– обґрунтування апаратної платформи для функціонування розподіленої комп’ютерної системи;

– проектування зручного користувацького інтерфейсу для введення даних про хворих за населеними пунктами, а також редагування їх властивостей;

– проектування та розробка програмного забезпечення для реалізації визначеної функціональності процесу збору та опрацювання статистичних показників захворюваності (desktop рішення);

– проектування схеми бази даних на основі реляційного підходу для зберігання та маніпулювання даними;

– реалізація алгоритмів пошуку інформації за визначеними критеріями;

– автоматизоване формування звітів визначеного зразка;

– формування графічної інтерпретації статистичних даних;

– зменшення затрат часу на виконання операцій при формуванні звітів.

Основні завдання, які покликана вирішити розподілена комп’ютерна система збору, аналізу та опрацювання даних щодо поширення захворюваності на COVID-19 полягає у формуванні бази даних статистичних показників, які повинні зберігатись на географічно-віддалених серверах. Для цього потрібно розробити схему бази даних, визначити вимоги до програмно-апаратних платформ для зберігання та доступу до інформації, побудувати архітектуру програмного забезпечення та реалізувати алгоритми опрацювання статистичних даних.

Окрім цього, кожен обласний лабораторний центр повинен мати можливість централізованого надсилання даних у глобальний репозиторій. У випадку надмірного завантаження таких центрів, потрібно забезпечити можливість запису результатів тестування з одного лабораторного центру до бази даних іншого. Тобто, крім основної функціональності, необхідно розробити ролі і права доступу до спільного використання баз даних між 24 обласними центрами.

Розподілена комп’ютерна система збору та аналізу даних щодо поширення вірусного захворювання повинна забезпечити підвищення ефективності роботи відділу епідеміологічного спостереження. Для цього необхідно проаналізувати його організаційну структуру, структуру вхідних документів, дані, які відображаються у звітах та розробити концептуальні схеми взаємодії та розподілу доступів до даних.

Основні цілі, яких потрібно досягти при використанні розподіленої комп’ютерної системи, полягають у підвищенні продуктивності праці як самих працівників, оскільки, зменшуються затрати часу на виконання операцій, так і підприємства в цілому. Крім того, при використанні такої системи збільшиться контроль над даними, їх актуальність та об’єктивність, а також в подальшому зростає достовірність і швидкість формування прогнозу захворюваності.

Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19 повинна забезпечувати доступ до інформації, що необхідна користувачам, згідно визначених прав доступу. Доступ до ресурсів повинен бути авторизованим та захищеним. Крім того, основною вимогою є автоматизоване формування звітів на основі вхідних даних.

До структури проектованої розподіленої комп’ютерної системи входить:

– апаратне та системне програмне забезпечення для функціонування комп’ютерної системи;

– база даних підтримки процесу формування та аналізу статистичних показників;

– клієнтська частина, що повинна забезпечити зв’язок та дружній інтерфейс між користувачами та базою даних;

– сервер обробки запитів;

У загальному випадку, модель бази даних повинна відображати предметну область, а клієнтська частина – відповідати за можливості обліку даних, забезпечення їх захисту, автоматичної генерації звітів.

Функціональні вимоги, що висуваються до розподіленої комп’ютерної системи:

– можливість введення, редагування та видалення даних щодо захворюваності за населеними пунктами та регіонами;

– можливість пошуку статистичних показників за вказаними критеріями;

– можливість сортування даних за визначеними критеріями;

– можливість запобігання неавторизованому доступу;

– можливість формування звітів;

– можливість керування правами доступу до інформаційних ресурсів;

– розподіл прав доступу;

– візуалізація звітних даних.

Перспективи розвитку системи включають перехід на іншу архітектуру бази даних, але при цьому її логічна структура не повинна змінюватися. Модернізація існуючої системи можлива при розширенні функціональності системи без кардинальних змін в існуючих схемах зв’язків між розподіленими компонентами та зв’язків між таблицями у базі даних.

У випадку виявлення або зміни сутностей предметної області, система повинна адекватно реагувати на внесення змін без пошкодження існуючих даних, а також повинні бути наявні механізми та інструменти резервного копіювання та відновлення даних.

Розподілена комп’ютерна система повинна бути захищена на рівні операційної системи сервера та авторизованого доступу до бази даних. Надійність системи повинна забезпечуватись також і у випадку збою роботи апаратного забезпечення.

Вимоги до функцій та задач, які виконує комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19, включають:

– забезпечення зв’язку клієнтської частини з базою даних;

– надання точних та адекватних результатів на запит користувачів;

– забезпечення часової ефективності роботи комп’ютерної системи;

– забезпечення контролю над доступом до інформації;

– забезпечення зручності використання апаратного і програмного забезпечення комп’ютерної системи;

– можливість розгортання та формування резервних копій бази даних.

Вимоги до апаратного забезпечення розподіленої комп’ютерної системи збору та аналізу даних передбачають вимоги до сервера та клієнтських робочих станцій.

Вимоги до сервера:

– процесор – тактова частота не менше 2,4 ГГц з кількістю логічних ядер не менше 8;

– об'єм оперативної пам'яті - не менше 16 Гб;

– об'єм жорсткого диску - не менше 1 Тб.

Вимоги до робочих станцій:

– процесор – тактова частота не менше 2,1 ГГц з кількістю логічних ядер не менше ;

– об'єм оперативної пам'яті - не менше 8 Гб;

– об'єм жорсткого диску – не менше 240 Гб.

Периферійні пристрої:

– принтер HP 1010 – 2 шт.

Програмне забезпечення комп’ютерної системи передбачає використання сервера на базі операційної системи Windows Server 2019.

Програмне забезпечення робочих станцій – Windows 10 та прикладне програмне забезпечення, що використовується для забезпечення необхідного рівня продуктивності праці.

1.2 Аналіз потенційних та можливих рішень щодо проектування розподілених комп’ютерних систем

Оскільки при організації будь-яких розподілених комп’ютерних систем найбільш критичними та важливими вимогами є вимоги часової ефективності, безпеки та функціональної повноти опрацювання даних, тому потрібно провести

аналіз підходів до проектування розподілених сховищ (баз даних), що забезпечують зберігання та маніпулювання такого роду інформацією.

В залежності від механізмів розподілу даних за вузлами мережі, типу апаратного і програмного забезпечення, а також процедур логічного поділу та формування запитів і відповідей сервера, можливе застосування таких підходів, як фрагментація відношень та реплікації баз даних або відношень.

Перед тим, як перейти до аналізу типів фрагментації та способів організації реплікацій дамо коротку характеристику видам розподілених систем керування базами даних. Розрізняють однорідні та неоднорідні розподілені СКБД.

Для однорідних РСКБД характерною особливістю є те, що на усіх розподілених вузлах комп'ютерної системи передбачається застосування систем керування базами даних однакового класу (для прикладу всі реляційні, або всі нереляційні). В ідеальному випадку, РСКБД на усіх вузлах функціонують на однаковому апаратному забезпеченні та з однаковим системним програмним забезпеченням.

Характерною особливістю неоднорідних систем керування базами даних є те, що вони функціонують на різних програмно-апаратних платформах та використовують різні типи СКБД, які відповідно реалізують різні моделі даних. До недоліків неоднорідних СКБД можна віднести той факт, що вони вимагають реалізації додаткових механізмів для синхронізації та перетворення даних з однієї моделі в іншу. Як наслідок, зростає вартість проектування та обслуговування розподілених комп'ютерних систем.

Виходячи з переваг і недоліків однорідних та неоднорідних типів СКБД, оптимальним рішенням для проектування та реалізації розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 є вибір однорідних реляційних систем керування базами даних, що в подальшому дозволить гнучко змінювати та застосовувати механізми фрагментації і реплікації таблиць бази даних.

Окрім однорідності програмно-апаратних платформ важливим є також забезпечення прозорості доступу до даних – властивість системи, що описує логічну незалежність даних від місця їхнього зберігання [1]. З практичної точки зору це означає, що користувачу буде забезпечено можливість сприйняття даних у межах його дозволів як єдиного цілого, а розподіл за вузлами системи не впливає на функціональну придатність системи.

Поділ бази даних за фрагментами переслідує ціль ефективного використання даних у тих вузлах комп'ютерної системи, де вони найбільш необхідні та не вимагає значних апаратних ресурсів для зберігання інформації. В якості фрагменту може розглядатись як окреме конкретне реляційне відношення, так складене.

У випадку розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 доцільно фрагмент представляти як відношення, що формується внаслідок виконання запиту до бази даних на локальному вузлі з подальшим його трансфером на центральний обчислювальний вузол або координатор.

Ще одним можливим варіантом організації розподіленої системи також є побудова однакових схем баз даних на локальних та глобальному вузлі, а логіка роботи програмного забезпечення комп'ютерної системи забезпечує одержання даних з усіх вузлів, їх цілісність і здатність до маніпулювання в поточний момент часу.

Якщо розглядати перший варіант, то в даному випадку, фрагмент представляє собою результат виконання запиту на деякому вузлі географічно-розподіленої системи. Основне завдання, яке при цьому потрібно розв'язати, полягає в ефективному застосуванні принципів фрагментації до безпосередньо відношень або по-іншому таблиць бази даних.

Комплексний критерій ефективності є архіважливим при відображенні фрагментів реляційного відношення і включає в себе наступні показники:

– час звернення і відповіді локального вузла,
– розмір пам'яті, необхідний для виконання запиту;

– кількість одночасних звернень до вузла (навантаження) та ряд інших.

Критеріями якості при формуванні фрагментації таблиць реляційної бази даних виступають повнота, відсутність перетинів та здатність до реконструювання. Ці критерії в комплексі описують інтегральну характеристику – коректність фрагментації.

Повнота фрагментації передбачає ситуацію, коли будь-який елемент даних реляційного відношення завжди відноситься до деякого фрагменту.

Якщо один і той самий елемент даних одночасно відноситься до кількох фрагментів реляційного (реляційних) відношення, то така фрагментація містить перетин, в іншому випадку – фрагментація не містить перетинів.

У випадку застосування деякої операції до фрагментів реляційного відношення як наслідок одержують початкове реляційне відношення, то схема фрагментації володіє здатністю до реконструювання.

Щодо типів фрагментації, то їх є три види. Перший вид – горизонтальна фрагментація передбачає, що схема фрагментів відношень є однаковою на усіх вузлах розподіленої комп'ютерної системи, а рядки чи записи в таблиці відрізняються і формуються на основі застосування операції селекції або обмеження. На рис. 1.1 наведено приклад горизонтальної фрагментації

Приклади горизонтальної фрагментації приведено на рис. 1.1 та рис. 1.2.

ID_Song	Title
1	Hey Jude
2	Here Comes the Sun
3	Something
4	Come Together

Рисунок 1.1 – Елементи даних при застосуванні горизонтальної фрагментації за умовою ID_Song<5

ID_Song	Title
5	Eleanor Rigby
6	While
7	My Guiltar Gently Weeps
8	Zhuravli
9	Zalizna lastivka
10	Sam sobi ostriv
11	Tak malo tut tebe
12	911
13	Ty sobi sama
14	Vesna
15	Yura
16	Test_Song

Рисунок 1.2 – Елементи даних горизонтальної фрагментації при ID_Song>=5

У випадку застосування операції об’єднання для всіх фрагментів утворених за допомогою операції селекції одержують відновлене вхідне відношення, тобто горизонтальна фрагментація відповідає критерію повноти [2]

$$R = \cup_{i=\overline{1,k}} \sigma_{F_i}(R) \tag{1.1}$$

R – таблиця, що підлягає фрагментації;

σ_{F_i} – сукупність умов, на основі яких виконується фрагментація.

Існує ще один вид горизонтальної фрагментації – породжена горизонтальна фрагментація. Даний вид фрагментації виникає у тому випадку, коли між таблицями існує тип зв’язку «один-до-багатьох» і при цьому в закінчення один з’являється породжена фрагментація, яка переходить у закінчення «багато».

У випадку наявності породженої горизонтальної фрагментації, доцільним є розташування пов’язаних фрагментів на одному і тому ж вузлі розподіленої комп’ютерної системи, оскільки зростає часова ефективність виконання запитів для цих фрагментів.

Інший вид або підхід до проектування та реалізації комп'ютерних систем, що містить фрагменти бази даних, є вертикальна фрагментація. Суть даного виду фрагментації полягає у розбитті (декомпозиції) реляційного відношення шляхом застосування операції проекції, тобто поділу таблиці за атрибутами. Особливістю такої декомпозиції є те, що атрибут-ідентифікатор вихідного реляційного відношення повторюється у всіх фрагментах, а за допомогою операції з'єднання за рівністю значень первинних ключів можна одержати вихідне реляційне відношення.

У випадку застосування індексів для певних наборів атрибутів також можлива реалізація вертикальної фрагментації.

Для прикладу, нехай маємо таблицю (реляційне відношення), екземпляр схеми якої наведено на рис. 1.3.

ID_Teacher	Surname	Position	Tel	ID_Department
12	Lutskiv A.M.	Assoc. Prof.	12345	15
13	Lupenko S.A.	Prof.	12345	15
14	Osukhivska H.M.	Assoc. Prof	12345	15
15	Lutsyk N.S.	Assoc. Prof	12345	15
16	Zharovskyy R.O.	Senior Teacher	12345	15
17	Shabliy N.R.	Assistant	12345	15
18	Sidorov M.V.	Prof	56789	16
19	Petrov O.V.	Assoc. Prof	56789	16
20	Ivanov S.V.	Assoc. Prof	56789	16
21	Pavlov S.S.	Assistant	56789	16

Рисунок 1.3 – Екземпляр схеми вихідної таблиці

У даному випадку прикладі, вертикальну фрагментацію запропоновано виконати за двома фрагментами. У першому фрагменті буде міститись інформація про ідентифікатор викладача (ID_Teacher), прізвище (Surname) та ідентифікатор кафедри (ID_Department), а в іншому – ідентифікатор викладача

(ID_Teacher), посада (Position) та номер телефону (Tel). У результаті проведення фрагментації, одержимо результат, як показано на рис. 1.4 та рис. 1.5.

ID_Teacher	ID_Department	Surname
12	15	Lutskiv A.M.
13	15	Lupenko S.A.
14	15	Osukhivska H.M.
15	15	Lutsyk N.S.
16	15	Zharovskyy R.O.
17	15	Shabliy N.R.
18	16	Sidorov M.V.
19	16	Petrov O.V.
20	16	Ivanov S.V.
21	16	Pavlov S.S.

Рисунок 1.4 – Перший фрагмент відношення

ID_Teacher	Position	Tel
12	Assoc. Prof.	12345
13	Prof.	12345
14	Assoc. Prof	12345
15	Assoc. Prof	12345
16	Senior Teacher	12345
17	Assistant	12345
18	Prof	56789
19	Assoc. Prof	56789
20	Assoc. Prof	56789
21	Assistant	56789

Рисунок 1.5 – Другий фрагмент відношення

Коректність вертикальної фрагментації означується через операцію об'єднання за проекціями атрибутів реляційного відношення, які відповідають утвореним фрагментам.

$$R = \bigcup_{i=\overline{1,n}} A_{R_i} \tag{1.2}$$

R – таблиця над якою виконують вертикальну фрагментацію;

A_{R_i} – сукупність проекцій таблиці, які утворюють її фрагменти.

До переваг вертикальної фрагментації відноситься:

– здатність реалізувати декомпозицію таблиці з таким розміщенням фрагментів за вузлами комп'ютерної системи, де вони найчастіше використовуються;

– можливість організації паралельного доступу та маніпулювання даними, що зберігаються у вертикальних фрагментах;

– підвищення ефективності процесу з'єднання фрагментів на основі ідентифікаторів.

У випадку одночасного використання горизонтальної та вертикальної фрагментації одержують ще один вид фрагментації – змішаний. На рис. 1.6 показано приклад змішаної фрагментації реляційних відношень, що передбачає послідовне виконання двох попередніх видів фрагментації.

Критеріями вибору того чи іншого виду фрагментації при реалізації розподілених комп'ютерних систем є наступні:

– швидкість обміну інформацією;

– час виконання запитів і формування відповідей;

– ціна зберігання даних на вузлах розподіленої комп'ютерної системи;

– існуючі обмеження щодо використовуваних ресурсів (об'єм пам'яті, час використання процесора та ін.).

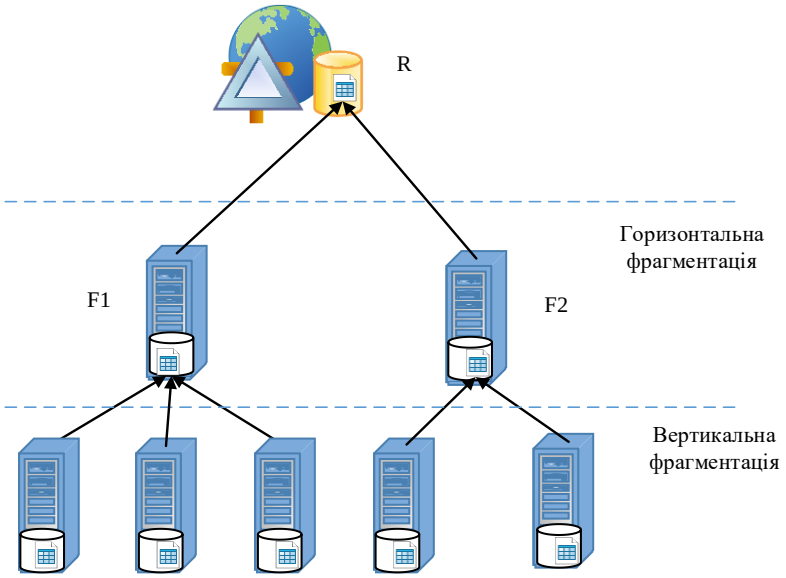

Рисунок 1.6 – Приклад організації змішаної фрагментації

Вхідними даними для побудови розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 є:

– кількість, тип та розмір таблиць, які планується розгорнути на локальних вузлах розподіленої системи;

– функціональні особливості щодо забезпечення процесу збору та аналізу даних;

– кількість і частота вихідних запитів для агрегування інформації у центральному вузлі розподіленої комп’ютерної системи;

– вимоги до апаратного забезпечення і вартості передачі даних, що забезпечують комунікацію між вузлами комп’ютерної системи.

При проектуванні та реалізації розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 запропоновано скористатись вертикальною фрагментацією та можливістю інтеграції горизонтальної декомпозиції таблиць, які розташовуються у локальних вузлах. Окрім цього, необхідно спроектувати архітектуру розподіленої комп’ютерної системи, забезпечити ефективність застосування методів одночасного доступу до даних на різних вузлах системи та синхронізацію і агрегацію даних на центральному вузлі системи.

РОЗДІЛ 2 ПРОЕКТУВАННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ ЗБОРУ ТА АНАЛІЗУ ДАНИХ ЩОДО ЗАХВОРЮВАНОСТІ НА COVID-19

2.1 Обґрунтування механізмів реплікації при проектуванні комп'ютерної системи

При проектуванні розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 доцільним є використання механізмів реплікації реляційних відношень, що дозволяє виконувати декомпозицію даних на відповідних вузлах. Відносно кількості вузлів за якими розподіляється система, то вона відповідає кількості обласних лабораторних центрів і становить 24 одиниці.

При застосуванні підходу реплікацій реалізується копіювання даних на вузлах розподіленої системи, що дає змогу збільшити швидкість виконання запитів та підвищити безвідмовність роботи системи [2]. Якщо застосовувати копіювання лише деяких реляційних відношень на локальних вузлах системи, то виконується неповна реплікація, а в іншому випадку – створюються копії усіх таблиць бази даних і така реплікація є повною.

У кваліфікаційній роботі бакалавра при розробці розподіленої системи збору та аналізу даних запропоновано використання неповної реплікації бази даних з можливістю інтеграції змішаної фрагментації. Перевагою такого підходу є динамічне створення БД розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19, а також синхронізації роботи локального і глобального обчислювальних вузлів системи. Як наслідок застосування такого підходу забезпечується ефективність прозорого доступу до даних та зростання якості виконання бізнес-процесів при формуванні статистики захворюваності пандемічним вірусом.

Базова схема реплікації реляційних відношень, в загальному випадку, наведена на рис. 2.1.

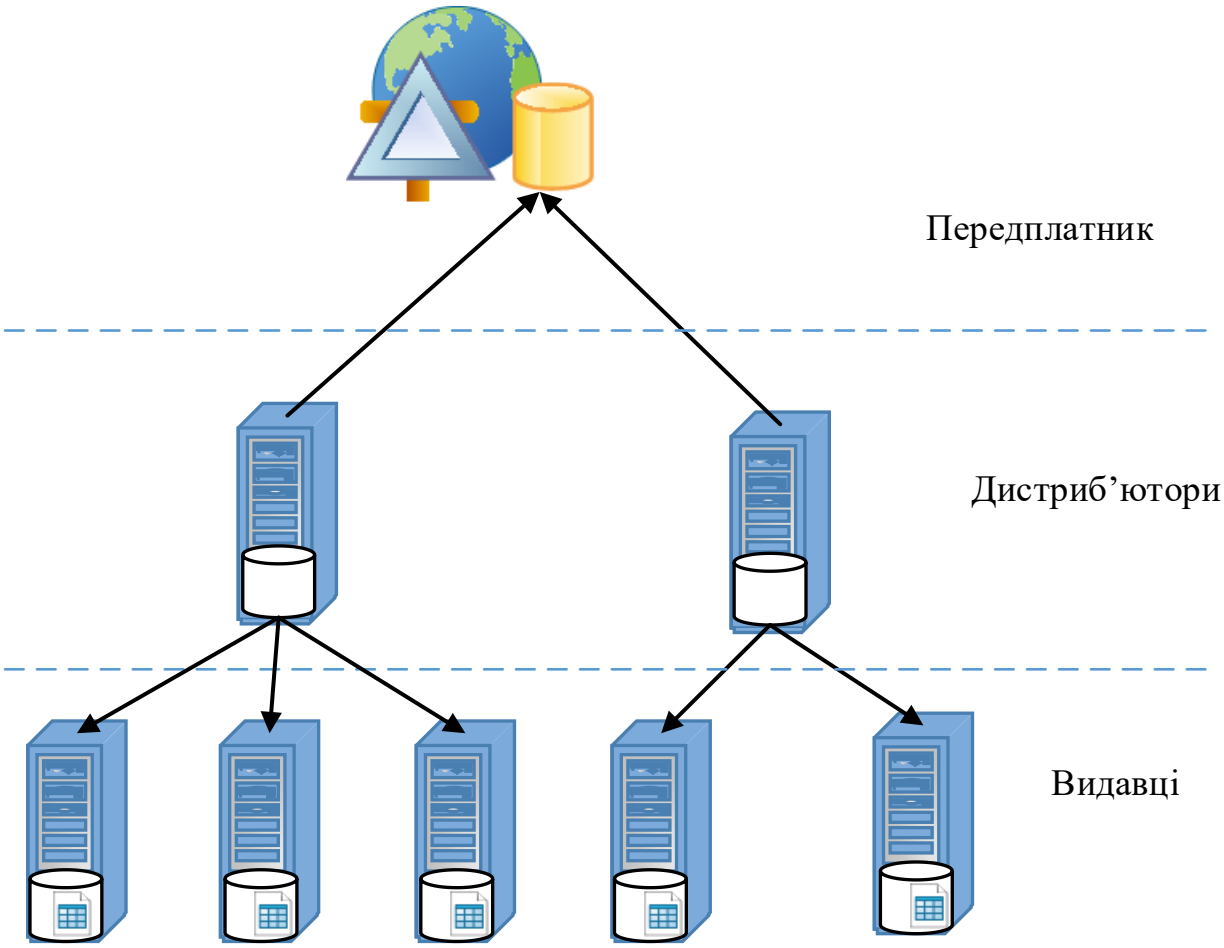

Рисунок 2.1 – Базова структура організації реплікації

Як видно з рис. 2.1 для реалізації механізму реплікації необхідно наявність трьох компонентів:

– видавець;
– дистриб'ютор;
– передплатник.

Основними функціями видавця, у випадку організації розподіленої комп'ютерної збору та аналізу даних, є збір інформації на локальних вузлах системи, що реалізуються шляхом введення та маніпулювання даних із застосуванням локального користувацького інтерфейсу.

Окрім цього, видавець забезпечує спостереження за внесенням змін у реляційних відношеннях і формування копій таблиць бази даних для подальшого транспортування на центральний глобальний вузол розподіленої системи.

До основних функцій дистриб'ютора входить підготовка і трансфер даних, які сформував видавець, на центральний глобальний вузол комп'ютерної системи. При цьому важливу роль відіграє тип організації методів реплікації, тобто дистриб'ютор може бути налаштований або на основі видавця, або на стороні передплатника.

За глобальне зберігання та опрацювання даних відповідає передплатник. Зазвичай, він представляє собою центральний глобальний вузол комп'ютерної системи, організований на основі хоста-сервера. На сервері мітиться глобальна схема бази даних і механізми підтримки та забезпечення цілісності даних щодо захворюваності на COVID-19.

На практиці, за способом організації, використовують наступні види реплікації:

– за запитом;
– примусова.

У випадку використання реплікації за запитом, центральний глобальний вузол, згідно деякого розкладу, формує запити до локальних вузлів розподіленої системи на предмет внесення змін у БД. Якщо зміни були внесенні за період між запитами, то передплатник одержує копії нових або змінених даних.

При застосуванні примусової реплікації, алгоритм забезпечення цілісності бази даних передбачає взаємодію дистриб'ютора, який визначає зміну стану БД і самостійну відправку змінених даних від компонентів-видавців до передплатника.

При проектуванні розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 доцільно застосовувати примусову реплікацію, оскільки це дозволить ефективно використовувати апаратні ресурси і завжди оперувати актуальними даними у будь-який момент часу.

При реалізації реплікації можливе застосування однієї з двох моделей – моментальні знімки бази даних та реплікація транзакцій [3].

У випадку застосування реплікації моментальних знімків, виконується повне копіювання схеми бази даних разом із самими даними на стороні видавців. Після цього інформація пересилається до передплатника (передплатників).

З точки зору оптимальності та ефективності використання ресурсів при організації розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності COVID-19 дана модель є непродуктивною, оскільки зростає завантаженість каналів зв'язку та вимагаються значні об'єми ресурсів для опрацювання знімків на стороні передплатника.

При організації моделі реплікації транзакцій, алгоритм формування реплік (копій) передбачає фіксацію у журналі дистриб'ютора послідовності виконання транзакцій на стороні видавців. Далі зафіксовані транзакції передаються до передплатника, що дозволяє зменшити об'ємів ресурсів для опрацювання даних та розвантажити канали зв'язку у розподіленій комп'ютерній системі.

Способи організації топологій розподілених комп'ютерних систем формуються по аналогії до побудови зв'язків між таблицями реляційних баз даних і передбачають наступні типи зв'язків між елементами розподіленої системи:

– «1 до ∞» – передбачає наявність одного видавця і кількох передплатників;

– «∞ до 1» – характеризується наявністю багатьох видавців і лише одного передплатника;

– «∞ до ∞» – компонентами розподіленої системи є більше, ніж 1 видавець і передплатник;

Перевагами організації реплікацій реляційних відношень є:

– здатність використання даних незалежно від типу вузла комп'ютерної системи;

– одночасне та/або паралельне формування запитів і відповідей між компонентами розподіленої системи;

– менша кількість використовуваних апаратних ресурсів при передачі даних, у порівнянні з іншими способами організації розподілених комп’ютерних систем.

До недоліків реплікації відносять:

– вища вартість виконання операцій над даними;

– зростання вимог до апаратних характеристик використовуваного обладнання;

– підвищення складності опрацювання запитів при задоволенні вимог щодо цілісності баз даних та мінімізації надлишковості даних.

Тому при проектуванні розподіленої комп’ютерної системи збору та аналізу даних варто використовувати наступні методи і механізми:

– змішана фрагментація реляційних відношень на розподілених вузлах комп’ютерної системи;

– імплементація моделі реплікації транзакцій;

– спосіб організації топології розподіленої комп’ютерної системи – «∞ до 1»;

– застосування трирівневої архітектури розподіленої системи, до складу якої повинні входити видавці, дистриб’ютори та передплатник.

Визначивши основні вимоги щодо проектування розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19, необхідно спроектувати її архітектуру на концептуальному і фізичному рівні.

2.2 Проектування моделі архітектури комп’ютерної системи

Концептуальна архітектура розподіленої комп’ютерної системи збору та аналізу даних щодо поширення COVID-19 повинна відображати кількість вузлів системи, їх типи, включати змішану фрагментацію та реплікацію реляційних відношень за вузлами системи, типи зв’язків як між апаратним, так і програмним забезпеченням. Окрім цього, запропоновано використовувати механізми управління даними на основі журналювання транзакцій, які повинні

виконуватись на локальному та глобальному центральному вузлі комп'ютерної системи.

Враховуючи те, що база даних захворюваності створюється на основі внесення інформації на локальних вузлах комп'ютерної системи, виникає необхідність її агрегації на центральному вузлі. Для вирішення цієї задачі передбачено компонент, який дозволяє трансформувати отримані дані чи транзакції з відповідного журналу, на структуру БД щодо захворюваності на COVID-19. Основною функцією такого механізму є агрегація статистичних показників за регіонами, місцем праці хворого, кількості захворюваності на визначеній території та інші. На рис. 2.2 наведено концептуальну архітектурну схема щодо збору та аналізу даних.

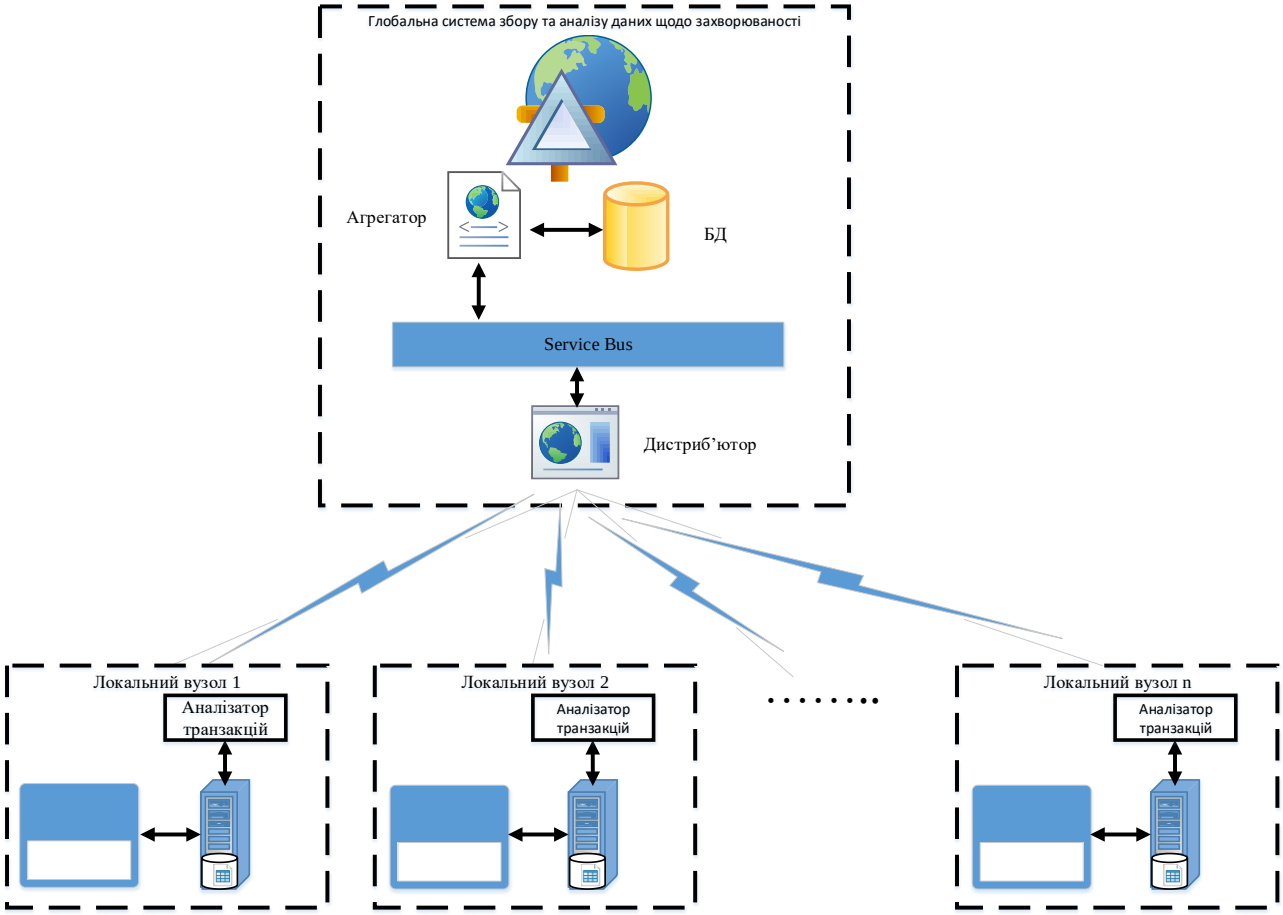

Рисунок 2.2 – Концептуальна архітектура системи збору та аналізу інформації щодо захворюваності на COVID-19

Формування бази даних щодо захворюваності на COVID-19 може відбуватися двома шляхами, в залежності від використовуваної точки входу користувача. При записі даних у базу даних на стороні глобального вузла, процедура і алгоритм роботи наступний.

Компонент Service bus призначений для формування та опрацювання черг при передачі даних з локальних вузлів системи на глобальний вузол.

У випадку зміни інформації у базі даних на глобальному вузлі, екземпляри схеми бази даних чи транзакції з використанням агрегатора надходять до Service bus. Оскільки, даний компонент системи відіграє роль балансувальника навантаження на вузол з можливістю паралельного виконання транзакцій, то втрат даних не повинно відбуватись за жодних обставин.

Одержавши дані, управління від Service bus передається дистриб'ютору. Дистриб'ютор, як було зазначено раніше, реалізує алгоритм примусової реплікації змінених або нових даних. Після цього він здійснює відправлення транзакцій на вказані локальні вузли комп'ютерної системи збору та аналізу даних.

До компонентів розподіленої комп'ютерної системи входять аналізатори транзакцій, які розташовані у кожному локальному вузлі системи. Основними їхніми функціями є аналіз, журналювання та оновлення стану виконання транзакцій, які відбуваються на тому чи іншому вузлі системи. У випадку звернення дистриб'ютора до видавця (аналізатора), даний компонент виконує трансфер журналу транзакцій у відповідний вузол системи, який сформував запит на одержання даних.

Процедура запису, оновлення чи редагування даних з локальних вузлів розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 виконується у зворотному напрямку.

Перевагами такої організації розподіленої системи є те, що маніпуляції над даними можна виконувати з будь-якого вузла за наявності відповідних прав доступу користувача.

2.3 Аналіз предметної області та виявлення інформаційних потоків у розподіленій комп'ютерній системи

Оскільки, основою розподіленої комп'ютерної системи, в даному випадку, є розподілена база даних, то для визначення сутностей та бізнес-процесів, які необхідно реалізувати, варто визначити та провести аналіз потоків даних, які будуть циркулювати у системі. На рис. 2.3 наведено спроектовану схему інформаційних потоків у розподіленій комп'ютерній системі.

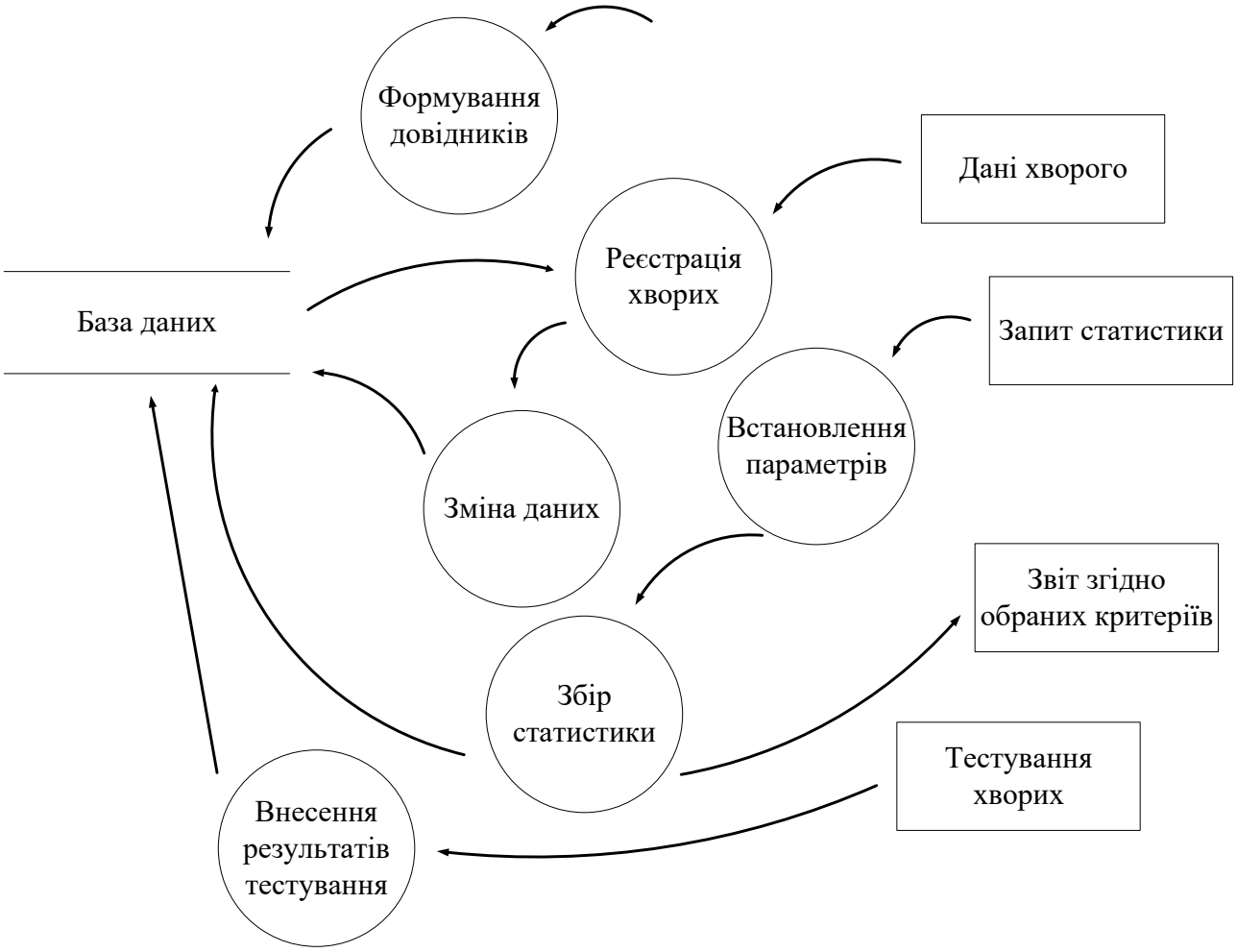

Рисунок 2.3 – Схема інформаційних потоків у розподіленій комп'ютерній системі

Основою функціонування проектованої системи є довідники, що містять загальну інформацію, наприклад, про населені пункти та регіони, де фіксують

захворюваність на COVID-19. Тому вони є важливою і невід'ємною компонентою системи, яка входить до складу бази даних.

Процес реєстрації особи, що проходить тестування на захворюваність пандемічним вірусом, включає інформацію про особисті дані пацієнта, а також дані з довідників про місце проживання та підприємство, де працює особа.

Даний процес пов'язаний із зміною стану бази даних, оскільки формуються нові записи або проводиться їхнє редагування.

Формування статистики виконується із застосуванням певних фільтрів в розрізі, наприклад, вікової категорії хворих, населеними пунктами, районами чи областями. Статистичні дані є основою при формуванні фільтрів, зборі та аналізі тенденції щодо поширення вірусу.

На основі зібраної статистики формуються звіти певного зразка та вигляду. Вони можуть бути представлені у вигляді табличної форми або у вигляді графіків.

Процес внесення результатів тестування передбачає фіксацію показників проведеного аналізу щодо особи, яка має підозру на хворобу.

Інформація щодо ефективної реалізації бізнес-процесів зберігається у фрагментах та репліках згідно архітектури розподіленої комп'ютерної системи, яка наведена на рис. 2.2.

Таким чином, проаналізувавши процеси та інформаційні потоки, які характерні для розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19, необхідно визначити і деталізувати наступне:

– сутності та атрибути предметної області, які є важливими для формування статистичних показників;
– користувачів системи разом з правами доступу і ролями;
– основні функціональні вимоги до системи;
– архітектуру програмного забезпечення розподіленої системи;
– алгоритм роботи програмного забезпечення;
– середовище імплементації програмного забезпечення та бази даних.

2.4 Визначення сутностей та атрибутів предметної області при проектуванні розподіленої комп'ютерної системи

Доцільність створення комп'ютерних систем чи будь-яких інших продуктів інформаційних технологій визначається задачами і майбутніми властивостями, які будуть реалізованими у системі і є цінними для кінцевих користувачів або власників системи.

Процес формулювання вимог і задач, які повинна розв'язувати розподілена комп'ютерна система збору та аналізу даних щодо захворюваності на COVID-19 є нетривіальним завданням. В загальному випадку, формування вимог щодо функціональності системи виконується шляхом збору явних вимог, тобто переліку потреб, які висловили користувачі системи. Однак існують ще приховані вимоги, які потрібно виявити на основі відомих потреб. Окрім цього, складністю процесу аналізу предметної області та визначення вимог є те, що розробники доволі часто не розуміють термінологічних та лексичних значень сутностей предметної області, а також зв'язків між ними.

Для встановлення явних та неявно висловлених потреб при проектуванні і реалізації комп'ютерних систем виконується процес аналізу предметної області, у результаті якого можна одержати сутності та атрибути предметної області, а також імовірні шляхи побудови моделей бізнес-процесів.

Для систематизації збору інформації про великих організаціях та подальшої розробки систем, що підтримують їх діяльність, застосовуються різноманітні способи схематичного представлення інформації про предметну область (схеми, таблиці, діаграми).

На основі аналізу предметної області щодо збору та аналізу захворюваності на COVID-19 виявлено сутності предметної області та їх атрибути, що є актуальними при розробці розподіленої комп'ютерної системи формування статистичної і звітної інформації. Для систематизації даних про сутності та атрибути отриману інформацію представлено у вигляді таблиць (табл. 2.1 – 2.14).

Таблиця 2.1 – Сутність «Частина країни»

Атрибут	Тип атрибута	Опис сутності
Назва	Текст	Сутність, що відповідає за частину країни, де фіксується захворюваність. Може набувати значень: Північ, Південь, Захід, Схід
Опис	Текст	

Таблиця 2.2 – Сутність «Область»

Атрибут	Тип атрибута	Опис сутності
Область	Текст	Сутність, що описує область України, кількість населення та до якої частини країни відноситься.
Частина країни	Текст	
Кількість населення	Цілочисельний	
Опис	Текст	

Таблиця 2.3 – Сутність «Район»

Атрибут	Тип атрибута	Опис сутності
Назва	Текст	Сутність, що описує район деякої області України, кількість населення у районі
Область	Текст	
Кількість населення	Цілочисельний	
Опис	Текст	

Таблиця 2.4 – Сутність «Населений пункт»

Атрибут	Опис сутності	Опис сутності
Назва	Текст	Сутність, що описує населений пункт, що належить до деякого району, тип населеного пункту та кількість населення
Тип	Текст	
Район	Текст	
Кількість населення	Цілочисельний	

Таблиця 2.5 – Сутність «Пацієнт»

Атрибут	Тип атрибута	Примітка
Прізвище	Текст	Сутність, що описує особу, яка проходить тестування на COVID-19
Ім’я	Текст	
По-батькові	Текст	
Адреса	Текст	
Номер телефону	Текст	
Вік	Цілочислений	
Місце праці (Організація)	Текст	

Таблиця 2.6 – Сутність «Організація»

Атрибут	Тип атрибута	Примітка
Назва організації	Текст	Сутність, яка описує підприємство чи організацію, де працює пацієнт
Населений пункт	Текст	
Тип організації	Текст	
Адреса	Текст	
Власник	Текст	

Таблиця 2.7 – Сутність «Тест»

Атрибут	Тип атрибута	Примітка
Назва тесту	Текст	Сутність, що описує процес тестування, тип тесту, пацієнт і результат тестування
Тип тестування	Текст	
Результат тестування	Текст	
Пацієнт	Текст	
Дата тестування	Дата і час	

Таблиця 2.8 – Сутність «Тип тесту»

Атрибут	Тип атрибута	Примітка
Назва типу тесту	Текст	Сутність, що вказує на характеристику типу тесту для діагностики COVID-19
Опис	Текст	

Таблиця 2.9 – Сутність «Звіт»

Атрибут	Тип атрибута	Примітка
Назва	Текст	Сутність, що відповідає за формування звітів щодо тестування захворюваності
Дата формування	Дата	
Вид звіту	Текст	
Звітний період	Дата	

Таблиця 2.10 – Сутність «Деталі звіту»

Атрибут	Тип атрибута	Примітка
ДеталіЗвіту	Цілочисельний	Сутність, що розкриває деталі звіту щодо тестування за звітний період
Значення за період	Число із фіксованою крапкою	
Пацієнт	Текст	
Група збору статистики	Текст	
Тест	Текст	
Вид обстеження	Текст	
Звіт	Текст	
Коментар	Текст	

Таблиця 2.11 – Сутність «Звітні періоди»

Атрибут	Тип атрибута	Примітка
Назва звітного періоду	Текст	Сутність, що описує шаблони звітних періодів щодо захворюваності пандемічним вірусом
Дата початку періоду	Дата	
Дата завершення періоду	Дата	
Коментар	Текст	

Таблиця 2.12 – Сутність «Група статистики»

Атрибут	Тип атрибута	Примітка
Назва групи статистики	Текст	Сутність, що розкриває суть статистичних показників та фільтрів при формуванні статистичних показників
Статистичні показники	Текст	

Таблиця 2.13 – Сутність «Тип організації»

Атрибут	Тип атрибута	Примітка
Назва типу організації	Текст	Сутність, що є допоміжним довідником для підприємства/ організації
Форма власності	Текст	

Таблиця 2.14 – Сутність «Тип населеного пункту»

Атрибут	Тип атрибута	Примітка
Назва типу населеного пункту	Текст	Сутність, що є допоміжним довідником при описі підприємства/ організації
Опис	Текст	

Таблиця 2.15 – Сутність «Тип звіту»

Атрибут	Тип атрибута	Примітка
Назва типу звіту	Текст	Сутність, що є довідником при формуванні звітів щодо тестування
Коментар	Текст	

На основі визначених сутностей в подальшому необхідно спроектувати зв’язки між ними, а для фізичної реалізації – сформувати первинні і зовнішні ключі (розділ 3).

Проведений аналіз предметної області дозволив з’ясувати структуру процесів діяльності відділу формування статистичних показників для обласних лабораторних центрів, виділити основні сутності та атрибути предметної області. З метою деталізації предметної області необхідно з’ясувати зв’язки між сутностями та їх властивості.

2.5 Побудова та характеристика зв’язків між сутностями

За допомогою зв’язків між сутностями можна зобразити яким чином відбувається взаємодія між ними та правильно сформувати реалізацію інформаційних потоків між компонентами системи. Розрізняють зв’язки між окремими незалежними сутностями та зв’язки всередині самої сутності, так звані рекурсивні зв’язки.

Зазвичай, взаємодія між сутностями іменується і виражається за допомогою дієслів. Фраза у вигляді дієслова визначає деяке обмеження та правило зв’язку між сутностями. Обмежень щодо кількості зв’язків між одними і тими ж сутностями не існує, однак вони повинні мати різне змістове навантаження.

Розрізняють два види сутностей: залежні та незалежні. У випадку незалежної сутності, екземпляри її схеми не взаємодіють з іншими сутностями і, зазвичай, виконують функції довідників.

У випадку наявності зв’язку між сутностями, одна з них є батьківською, а інша – дочірньою. Відповідно для дочірнього екземпляру сутності повинен існувати батьківський екземпляр.

Розрізняють наступні типи зв'язків між сутностями:

– зв'язок «один до багатьох» – ідентифікує екземпляр батьківської сутності у дочірній шляхом штучного створення зовнішнього ключа, що є ідентифікатором у батьківській сутності (сутності є залежними між собою);

– факультативний зв'язок «один до багатьох» – даний тип зв’язку може бути встановлений між незалежними сутностями (екземплярами сутностей), а первинний ключ ключ батьківської сутності міститься у дочірній як неключовий атрибут;

– зв'язок «багато – до –багатьох» – фізично реалізується шляхом створення проміжної сутності і встановленням двох зв’язків типу «один-до-багатьох».

Для прикладу, сутність «ГрупаЗакладів» містить множину екземплярів «Об’єктНагляду», цей зв'язок характеризується кратністю «один до багатьох» та є обов’язковим для сутності «Об’єктНагляду». З боку залежної сутності «Об’єктНагляду» зв'язок підтримується наявністю ідентифікуючого атрибуту (зовнішнього ключа), що містить однозначне посилання на відповідний первинний ключ сутності «ГрупаЗакладів».

Рисунок 2.4 – Зв’язок між сутностями «ГрупаЗакладів» та «Об’єктНагляду»

Окрім цього, модель предметної області може містити і зв’язки, що вказують на приналежність сутності до сутностей того ж відношення – наприклад так, як певний регіон входить в склад іншого. Такий зв'язок називають

рекурсивним, він може утворювати нескінченну ієрархію. Наприклад, сутність «ГрупаЗборуСтатистики» може містити підгрупи або бути віднесеною до певної батьківської групи.

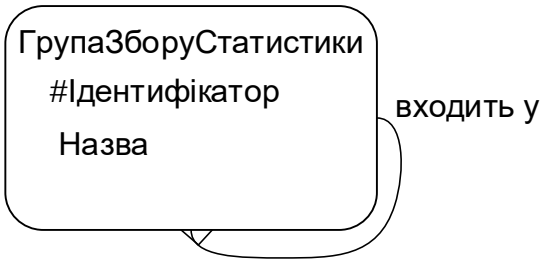

Рисунок 2.5 – Рекурсивний зв'язок сутності «ГрупаЗборуСтатистики»

Визначення сутностей предметної області та представлення прикладних задач, що повинні в ній виконуватись у вигляді зв’язків між сутностями та їх атрибутів дозволяє виконати проектування структури та фізичну реалізацію бази даних, яка приведена у розділі 3.

2.6 Визначення вимог та ролей у розподіленій комп’ютерній системі

Вимоги до розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 пропонується представити у вигляді діаграми варіантів використання, а опис ролей користувачів системи – у табличному вигляді.

В основі діаграм варіантів використання лежать так звані прецеденти, що описують деякий сценарій взаємодії користувача або системи з актором. При цьому такий прецедент повинен виконувати значущу функціональність у розподіленій комп’ютерній системі.

Засобами побудови діаграм варіантів використання є мова UML. Для позначення прецеденту використовується позначення іменованого овалу, при цьому він містить назву функції, що виражена за допомогою дієслова.

Користувачі системи позначаються у вигляді силуетів і можуть містити назву конкретного користувача, представляти різні ролі або зовнішні системи, які взаємодіють з конкретною системою.

Зв'язки між прецедентами і акторами, або між кількома прецедентами можуть бути представленими у вигляді наступних зв'язків:

– узагальнення (generalization) – узагальнюючий прецедент пов'язаний з прецедентами, які його деталізують;

– розширенням (extend relationship) – додаткові функціональні можливості, пов'язані з даним прецедентом;

– включенням (include relationship) – до складу даного прецеденту входять інші прецеденти.

Між акторами також можна будувати зв'язки узагальнюючого типу. На рис. 2.6 наведено USE CASE діаграму, що відображає функціональні вимоги, які необхідно реалізувати у розподіленій комп'ютерній системі збору та налізу даних щодо захворюваності на COVID-19.

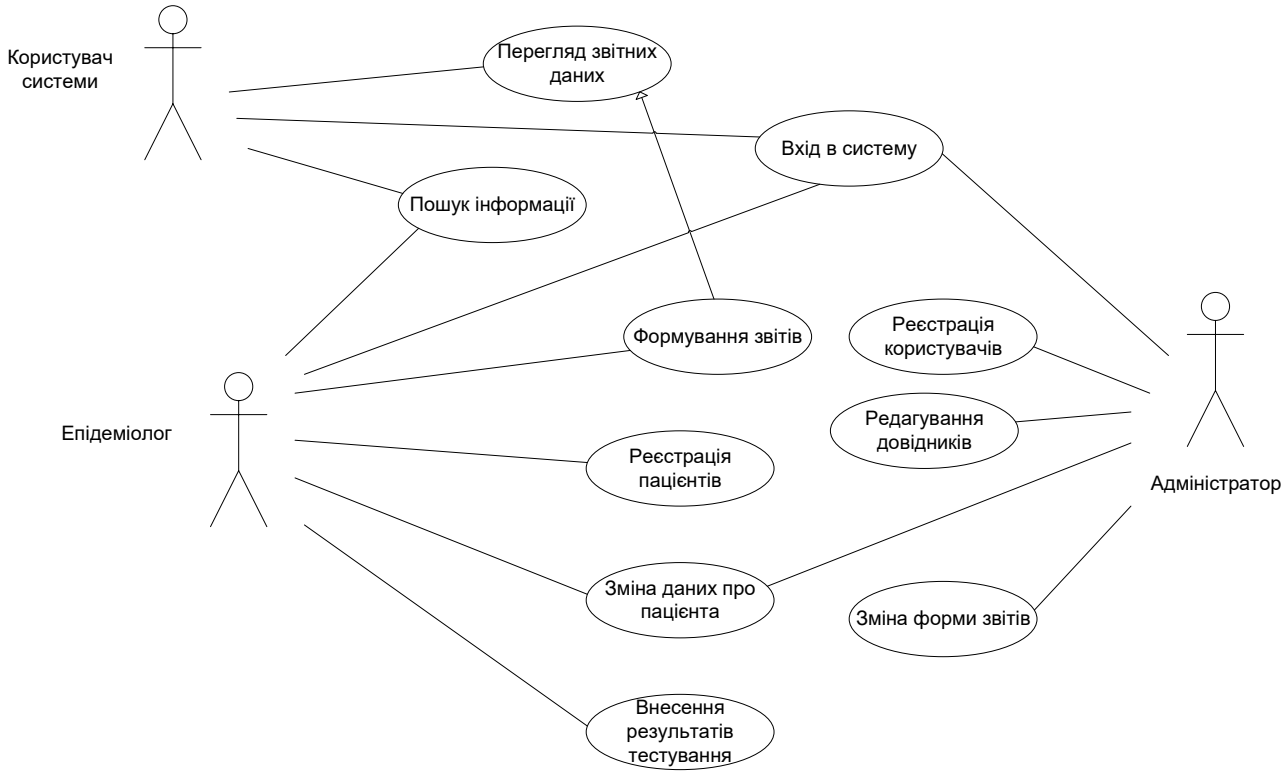

Рисунок 2.6 – Діаграма варіантів використання при проектуванні розподіленої комп'ютерної системи

Лікар-епідеміолог може сформувати звіт на основі певних даних, але якщо дані наперед не відомі, їх необхідно відібрати із допомогою засобів пошуку. Тому варіант використання «Формування звіту» розширює варіант використання «Перегляд звітних даних».

Зміна даних про підприємство може використовуватись тільки для уже існуючого підприємства та внесених для нього даних. Тому варіант використання «Зміна даних пацієнта» буде повністю включати сценарій «Реєстрація пацієнта».

Ролі користувачів та потенційні сценарії роботи з розпоідленою комп'ютерною системою наведено у табл. 2.15.

Таблиця 2.15 – Рольова модель користувачів комп'ютерної системи

Назва ролі	Опис ролі	Основні сценарії
Лікар-епідеміолог	Іспектуючі лікарі-епідеміологи проводять нагляд за проведенням тестування щодо захворюваності, фіксують результати тестування, формують відповідну звітну документацію	– реєстрація пацієнтів з підозрою на захворювання; – реєстрація підприємств, де працюють пацієнти; – зміна даних про пацієнтів та підприємства; – внесення результатів тестування; – фіксація статистичної інформації; – пошук та перегляд інформації; – формування звітів.
Користувачі	Користувачі системи, що мають дозвіл переглядати інформацію, але не є лікарями-епідеміологами	– пошук та перегляд звітної інформації

Продовження таблиці 2.15

Назва ролі	Опис ролі	Основні сценарії
Адміністратор системи	Привілейована роль в системі, проводить заповнення основних довідників та моніторинг дієздатності програних модулів	– реєстрація користувачів; – редагування даних; – формування звітів; – аналіз даних.

Діаграми варіантів використання дозволяють не тільки більш детально з'ясувати особливості функціонування певної предметної області, вони також є хорошим підґрунтям для формування функціонального каркасу прикладних додатків в яких сценарії реалізуються послідовністю екранів чи сторінок

На основі аналізу діяльності лабораторних центрів при формуванні звітності було розроблено діаграму варіантів використання, що відображає основні сценарії та прикладні задачі, які повинна підтримувати розподілена комп'ютерна система. Наступний крок полягає у розробці програмного забезпечення розподіленої комп'ютерної системи з інтеграцією запропонованих проектних рішень.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Проектування та реалізація архітектури програмного забезпечення локальних вузлів розподіленої системи

Для реалізації архітектури програмного забезпечення на локальних вузлах комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 пропонується використати архітектуру «клієнт-сервер». Реалізація архітектури на основі клієнт-серверної взаємодії передбачає конструювання кількох рівнів, що входять до складу ПЗ. Такі рівні визначають у відповідності до їх функціонального призначення:

– сховище (репозиторій) даних;
– рівень опрацювання даних;
– прикладний користувацький інтерфейс.

На рівні репозиторію, зазвичай, використовують підхід організації реляційних БД у відповідному СКБД, що володіє механізмами горизонтальної і вертикальної фрагментації.

Оскільки, сучасні РСКБД є досить розвинутими середовищами та надають зручні і потужні засоби керування базами даних, організація роботи сховищ інформації у випадку зі сценаріями багаторівневого масштабування є добре структурованою і керованою процедурою. Архітектура система процесу формування збору, аналізу і звітності щодо захворюваності пандемічним вірусом використовує модель спільного репозиторію даних, доступ до якого виконується віддаленими прикладними додатками по комп'ютерній мережі за допомогою рівня посередника.

Функціями інтерфейсу посередника клієнта є наступні:

– формування параметрів клієнтської сторони додатку для забезпечення можливості звернення до відповідного сервісу на сервері;

– безпосереднє відправлення сформованих параметрів сервера з можливістю подальшого його запуску для одержання результатів або повідомлень про збій.

До функцій інтерфейсу посередника на стороні сервера входять:

– одержання повідомлення від клієнтської сторони, старт процедури RPC, формування результату і препроцесинг (кодування або перекодування) даних у зрозумілому для клієнта вигляді;

– відправлення результату до клієнта, який сформував запит у форматі параметрів повідомлення та видалення RPC.

Реалізацію програмного забезпечення локальних вузлів розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності запропоновано виконати із застосуванням стеку технологій Microsoft. На рис. 3.1 наведено архітектуру програмного забезпечення, що буде використовуватися у комп’ютерній системі.

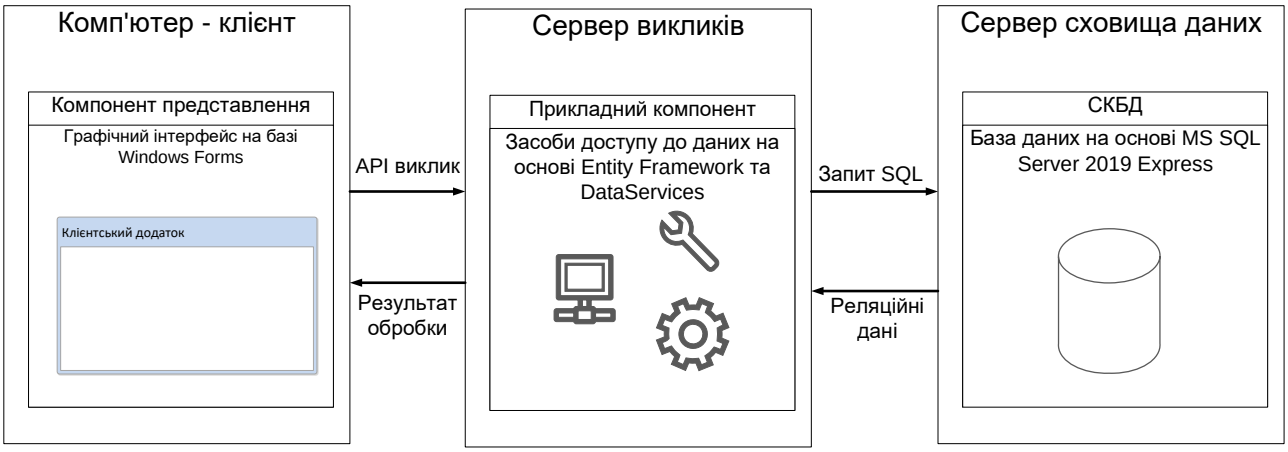

Рисунок 3.1 – Архітектура програмного забезпечення на локальних вузлах розподіленої комп’ютерної системи

У даному випадку, на клієнтській стороні (комп’ютер-клієнт) компонент представлення реалізовується із використанням технології Windows Forms.

Це дає змогу розробити зручний віконний користувацький інтерфейс.Сервер викликів побудований на основі засобів доступу до даних Entity Framework та DataServices. Entity Framework представляє собою систему ORM (Object-Relational Mapping), що забезпечує здатність маніпулювання даними на основі об’єктної моделі без використання логічної або реляційної схеми даних. Перевагою використання ORM є швидкість реалізації моделей класів для представлення предметної області, контроль коду щодо доступу до інформації та здатність гнучкого налаштування параметрів при міграції схеми БД.

При роботі зі сховищами, де дані зберігаються у вигляді таблиць, Entity Framework використовує LINQ To Entities, як альтернативний інтерфейс LINQ API при зверненні до БД. Цей механізм дозволяє розрізняти об'єктну модель сутностей від фізичної реалізації БД шляхом формування логічного відображення між ними.

З метою відображення одних типів даних на інші (наприклад, типів C#.NET в типи SQL) використовується незалежний від мов програмування декларативний опис. Найпоширенішим засобом, що для цього застосовується в сучасності є технологія XML/XSLT та базовані на них схеми представлення даних XSD/WSDL. При розробці системи збору та аналізу даних щодо захворюваності на COVID-19 для доступу до даних єдиного репозиторію запропоновано використати засоби ORM Entity Framework, що базуються на представленні моделі бази даних у вигляді схеми Entity Data Model (EDM), яка базується на нотації XSD.

Для створення бібліотеки роботи із даними з допомогою моделі Entity Framework CodeFirst необхідно на основі даних про сутності предметної області та їх атрибути, сформувати прості класи (т.з. Plain Old CLR Object, прості типи CLR). Наприклад, клас що відповідає сутності «Група закладів» може бути представлений кодом класу C#, який показано у лістингу 3.1.

Лістинг 3.1 – Програмний код класу CompanyesGroup

```
public class CompanyesGroup
    {
        public int CompanyesGroupId { get; set; }

        public string Name { get; set; }

        public int? ParentId { get; set; }

        [ForeignKey("ParentId")]
        public CompanyesGroup ParentGroup { get; set; }
    }
```

Кожна сутність POCO, що повинна відображатись на таблицю бази даних повинна задовольняти такі умови:

– бути простим класом, що не містить залежностей від класів платформи;

– усі властивості сутностей повинні бути описані з модифікатором доступу public;

– кожна сутність повинна містити ідентифікуючу властивість цілочисельного типу чи типу GUID.

Для того щоб можна було взаємодіяти з даними всередині таблиць бази даних використовується спеціальний клас, який входить до складу платформи Entity Framework – DbContext. Цей клас є спеціально розробленою версією контексту даних в основі якого використано технологію ADO.NET.

Для опису власної моделі роботи з даними необхідно створити клас, що буде наслідувати DbContext. Представлення таблиць в базі даних виконується з допомогою спеціального узагальненого типу DbSet<T>, який реалізує засоби доступу до таблиць в фізичній бази даних (лістинг 3.2).

Лістинг 3.2 – Клас наслідуваний від DbContext

```
public class IssDbContext : DbContext
    {
        public DbSet<Company> Companies { get; set; }

        public DbSet<WorkerGroup> WorkerGroups { get; set; }

        public DbSet<CompanyGroup> CompanyGroups { get; set; }
    }
```

Для організації слабко зв'язаної архітектури та з метою відділення засобів доступу до даних і користувацького інтерфейсу використовують шаблон проектування, що отримав назву «репозиторний шаблон». Полягає цей підхід в створенні спеціального рівня представлення – репозиторіїв, які інкапсулюють логіку роботи з даними в моделі одночасно не розкриваючи технічні деталі реалізації доступу до даних.

Для реалізації репозиторного шаблону використовують дворівневу модель класів. Спочатку розробляються абстрактні представлення (в випадку C# це інтерфейси), які декларують доступні методи роботи з даними (лістинг 3.3).

Лістинг 3.3 –Дворівнева модель класів

```
public interface IEntityItemsRepository
    {
        Guid SaveEntityItem(EntityItem itemToSave);

        IList<EntityItem> GetEntitiesList();

        IEnumerable<EntityItem> GetEntitiesForSchemaId(Guid schema
Id);

        EntityItem GetEntityById(Guid entityId);

        void DeleteEntity(Guid entityId);

        IEnumerable<EntityItem> SeachInSchema(string searchItem, G
uid SchemaId);

        IDictionary<Guid, IEnumerable<EntityItem>> SearchInAllData
base(string searchItem, IEnumerable<Guid> schemaIdList);

        void MoveElementUp(Guid entityId);

        void MoveElementDown(Guid entityId);

        IEnumerable<EntityItem> SearchItemLikeness(EntityItem item
);
    }
```

Наступним кроком є реалізація специфічного класу (лістинг 3.4), який наслідує репозиторний інтерфейс і використовує конкретну технологію доступу до даних – наприклад, Entity Framework.

Лістинг 3.4 – Наслідування репозиторного інтерфейсу

```
internal class EFElementsRepository : IElementsRepository
    {
        private IssDbContext context = new IssDbContext();

        #region IElementsRepository Members

        public IQueryable<Element> ElementsList
        {
            get { return context.Elements.Include(x => x.InSchema)
.Include(x => x.EntityPropertyValues); }
        }

        public Guid SaveElement(Element itemToSave)
        {
            if (itemToSave.ElementId == Guid.Empty)
            {
                itemToSave.ElementId = Guid.NewGuid();
                itemToSave.SortNumber = itemToSave.SortNumber == 0
 ? GetMaxSortNumber(itemToSave.SchemaId) + 1 : itemToSave.SortNumb
er;
                context.Elements.Add(itemToSave);
                savePropertiesValues(itemToSave, itemToSave);
            }
            else
            {
                var oldElement = GetElementById(itemToSave.Element
Id);
                oldElement.SortNumber = itemToSave.SortNumber != 0
 && itemToSave.SortNumber != oldElement.SortNumber ? itemToSave.So
rtNumber : oldElement.SortNumber;
                savePropertiesValues(itemToSave, oldElement);
            }
            context.SaveChanges();
            return itemToSave.ElementId;
        }

        private void savePropertiesValues(Element itemToSave, Elem
ent oldItem)
        {
            foreach (var propertyValue in itemToSave.EntityPropert
yValues)
            {
                var oldProp = context.EntityPropertyValues.FirstOr
Default(x => x.ElementId == oldItem.ElementId && x.PropertyId == p
ropertyValue.PropertyId);
                propertyValue.ElementId = oldItem.ElementId;
                propertyValue.Element = oldItem;
                if (oldProp == null)
                {
                    propertyValue.EntityPropertyValueId = Guid.New
Guid();
```

```
                    context.Entry(propertyValue).State = EntitySta
te.Added;
                }
                else
                {
                    oldProp.Value = propertyValue.Value;
                }
            }
        }

        public Element GetElementById(Guid elementId)
        {
            return ElementsList.FirstOrDefault(x => x.ElementId ==
 elementId);
        }

        public IQueryable<Element> GetElementsForSchemaId(Guid sch
emaId)
        {
            return ElementsList.Where(x => x.SchemaId == schemaId)
;
        }

        public void DeleteElement(Guid elementId)
        {
            var element = GetElementById(elementId);
            if (element != null)
            {
                context.Elements.Remove(element);
                foreach (var propertyValue in element.EntityProper
tyValues)
                {
                    context.EntityPropertyValues.Remove(propertyVa
lue);
                }
                context.SaveChanges();
            }
        }

        public int GetMaxSortNumber(Guid schemaId)
        {
            if (context.Elements.Count(x => x.SchemaId == schemaId
) == 0)
            {
                return 0;
            }
            return context.Elements.Where(x => x.SchemaId == schem
aId).Max(x => x.SortNumber);
        }

        #endregion IElementsRepository Members
    }
```

Використовуючи таку модель доступу до даних розробники можуть легко заміняти часткові реалізації методів репозиторію чи навіть повністю класи репозиторіїв не порушуючи логіки роботи з даними. При цьому рівень користувацького інтерфейсу (представлення) є повністю незалежним від конкретної технології доступу до даних.

Програмний код рівня представлення використовує декларативний тип інтерфейсу репозиторію замість виклику екземпляру конкретного класу. З метою надання доступу до можливостей репозиторію використовується шаблон «Фабрика», який полягає в розробці спеціалізованого класу призначеного для конструювання екземплярів класів-репозиторіїв (лістинг 3.5).

Лістинг 3.5 – Конструювання екземплярів класів-репозиторіїв

```
public class RepositoryFactory
    {
        public static ISchemaItemsRepository GetSchemaItemReposito
ry()
        {
            EFSchemasRepository schemasRep = new EFSchemasReposito
ry();
            EFElementsRepository elementsRep = new EFElementsRepos
itory();
            EFSchemaItemsRepository itemRep = new EFSchemaItemsRep
ository(schemasRep, elementsRep);
            return itemRep;
        }

        public static IEntityItemsRepository GetEntityItemsReposit
ory()
        {
            return new EFEntityItemsRepository(new EFElementsRepos
itory());
        }

        public static IDataTypesRepository GetDataTypesRepository(
)
        {
            return new EFDataTypesRepository();
        }
    }
```

Використання технології Entity Framework та репозиторного шаблону значно спрощує роботу з даними, запобігає необхідності написання специфічного SQL коду, забезпечує незалежність рівнів програмної системи.

Загальний алгоритм роботи програмної складової розподіленої комп'ютерної системи, а також алгоритм формування звітної документації наведено у графічному матеріалі КС КРБ 123.159.00.02 та КС КРБ 123.159.00.03 відповідно.

3.2 Реалізація бази даних у середовищі MS SQL Server 2019 Express

На основі визначених сутностей предметної області, представлених у табл. 2.1 – табл.2.15 у розділі 2, засобами мови SQL у середовищі MS SQL Server 2019 Express фізично реалізовано реляції бази даних, структура яких приведена у вигляді табл. 3.1 – 3.15.

SQL –запит на створення таблиці CountryPart наведено у лістингу 3.6, а її схему у вигляді табл. 3.1.

Лістинг 3.6 – SQL-запит на створення таблиці CountryPart

```
create table CountryPart
(
ID_Part int not null primary key identity (1,1),
Part_Name varchar (150),
[Description] varchar (MAX)
)
```

Таблиця 3.1 – «CountryPart»

Атрибут	Тип	Примітка
ID_Part	int	Primary key (PK)
Part_Name	varchar(150)	
Description	varchar(MAX)	

SQL –запит для формування таблиці District представлено у лістингу 3.7, а її структуру – у табл. 3.2.

Лістинг 3.7 – SQL-запит на створення таблиці Distric

```
create table District
(
ID_District int not null primary key identity (1,1),
ID_Part int foreign key references CountryPart (ID_Part),
District_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
```

Таблиця 3.2 – «District»

Атрибут	Тип	Примітка
ID_District	int	PK
ID_Part	int	Foreign Key (FK)
District_Name	varchar(150)	
People_Number	int	
Description	varchar(MAX)	

Запит для формування таблиці, що відповідає за район наведено у лістингу 3.8, а схема одержаної таблиці – у табл. 3.3.

Лістинг 3.8 – Створення таблиці «Region»

```
create table Region
(
ID_Region int not null primary key identity (1,1),
ID_District int foreign key references District (ID_District),
Region_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
```

Таблиця 3.3 – «Region»

Атрибут	Тип	Примітка
ID_Region	int	PK
ID_District	int	FK
Region_Name	varchar(150)	
People_Number	int	

Для зберігання інформації про населений пункт, де зафіксовано захворюваність на COVID-19, створено таблицю Locality (лістинг 3.9, табл. 3.4).

Лістинг 3.9 – Створення таблиці «Locality»

```
create table Locality
(
ID_Locality int not null primary key identity (1,1),
ID_Region int foreign key references District (ID_District),
ID_Type int foreign key references LocalityType (ID_Type),
Locality_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
```

Таблиця 3.4 – «Locality»

Атрибут	Тип	Примітка
ID_Locality	int	PK
ID_Region	int	FK
ID_Type	int	FK
Locality_Name	varchar(150)	
People_Number	int	
Description	varchar(MAX)	

У лістингу 3.10 наведено скрипт для створення таблиці-довідника щодо типу населеного пункту, а відповідно у табл. 3.5 – одержану схему таблиці.

Лістинг 3.10 – Скрипт створення таблиці LocalityType

```
create table LocalityType
(
ID_Type int not null primary key identity (1,1),
LType_Name varchar (50)
)
```

Таблиця 3.5 – «LocalityType»

Атрибут	Тип	Примітка
ID_Type	int	PK
LType_Name	varchar(50)	

Для зберігання інформації про місце праці особи, що проходить тестування на захворюваність COVID-19, передбачено таблицю «Organization». Скрипт для створення цієї таблиці приведено у лістингу 3.11, а її схему – у табл. 3.6.

Лістинг 3.11 – SQL-запит створення таблиці «Organization»

```
create table Organization
(
ID_Organization int primary key  not null identity (1,1),
ID_Locality int foreign key references Locality(ID_Locality),
ID_Org_Type int foreign key references Org_Type(ID_Org_Type),
Org_Name varchar (MAX),
[Address] varchar (max),
[Owner] varchar (150),
[Description] varchar (MAX)
)
```

Таблиця 3.6 – «Organization»

Атрибут	Тип	Примітка
ID_Organization	int	PK
ID_Locality	int	FK
ID_Org_Type	int	FK
Org_Name	varchar(MAX)	
Address	varchar(MAX)	
Owner	varchar(150)	

З метою деталізації інформації про підприємство, створено таблицю, що описує її тип і форму власності. У лістингу 3.12 і табл. 3.7 наведено відповідно скрипт генерації цієї таблиці і її схему.

Лістинг 3.12 – Створення таблиці «Org_Type»

```
create table Org_Type
(
ID_Org_type int not null primary key identity (1,1),
Org_Type_Name varchar (100),
Org_Form varchar (100)
)
```

Таблиця 3.7 – «Org_type»

Атрибут	Тип	Примітка
ID_Org_type	int	PK
Org_Type_Name	varchar(100)	
Org_Form	varchar(100)	

Для зберігання даних про особу, яка проходить тестування або має підозру на захворюваність COVID-19, передбачено реляційне відношення «Patient», SQL-запит реалізації якої наведено у лістингу 3.13, а схему – у табл. 3.8.

Лістинг 3.13 – SQL-запит на створення таблиці «Patient»

```
create table Patient
(
ID_Patient int not null primary key identity (1,1),
ID_Organization int foreign key references
Organization(ID_Organization),
FirstName varchar (100),
MiddleName varchar (150),
LastName varchar (100),
PhoneNumber varchar (20),
Passport varchar (15),
Age int
)
```

Таблиця 3.8 – «Patient»

Атрибут	Тип	Примітка
ID_Patient	int	PK
ID_Organization	int	FK
FirstName	varchar(100)	
MiddleName	varchar(150)	
LastName	varchar(100)	
PhoneNumber	varchar(20)	
Passport	varchar(15)	
Age	int	

Оскільки, тестування на захворюваність COVID-19 передбачає застосування різних типів тестів, тому передбачено таких два реляційних відношення як «Test» і «TestType», скрипт створення яких наведено у лістингу 3.14 та лістингу 3.15 відповідно, а їхні схеми у табл. 3.9-табл. 3.10.

Лістинг 3.14 – Створення реляційного відношення «Test»

```
create table Test
(
ID_Test int not null primary key identity (1,1),
Test_Name varchar (200),
TestValue int,
ID_TType int foreign key references TypeTest (ID_TTest),
ID_Patient int foreign key references Patient(ID_Patient),
ID_Parent_Test int foreign key references Test(ID_Test)
)
```

Лістинг 3.15 – Створення реляційного відношення «TypeTest»

```
create table TypeTest
(
ID_TTest int not null primary key identity (1,1),
[Type_Name] varchar (200),
)
```

Таблиця 3.9 – «Test»

Атрибут	Тип	Примітка
ID_Test	int	PK
Test_Name	varchar(200)	
TestValue	int	
ID_TType	int	FK
ID_Patient	int	FK
ID_Parent_Test	int	FK

Таблиця 3.10 – «TypeTest»

Атрибут	Тип	Примітка
ID_TTest	int	PK
Type_Name	varchar(200)	

Формування групи звітів щодо захворюваності здійснюється на основі реалізованої у лістингу 3.16 таблиці. Результат формування схеми БД наведено у табл. 3.11.

Лістинг 3.16 – Формування таблиці «Report»

```
create table Report
(
ID_Report int not null primary key identity (1,1),
ReportName varchar (max),
ID_R_Period int foreign key references ReportPeriod(ID_R_Period),
ID_RType int foreign key references ReportsType (ID_RType),
ReportDate datetime
)
```

Таблиця 3.11 – «Report»

Атрибут	Тип	Примітка
ID_Report	int	PK
ReportName	varchar(MAX)	
ID_R_Period	int	FK

Продовження таблиці 3.11

Атрибут	Тип	Примітка
ID_RType	int	FK
ReportDate	datetime	

При формуванні звітів використовується дві таблиці довідники – тип звіту та період за який необхідно сформувати звіт. У лістингу 3.17 та 3.18 відповідно наведено SQL-запит для формування цих таблиць, а їхні схеми – відповідно у табл. 3.12 і табл. 3.13.

Лістинг 3.17 – Створення таблиці «ReportsType»

```
create table ReportsType
(
ID_RType int not null primary key identity (1,1),
RType_Name varchar (200),
[Description] varchar (max)
)
```

Лістинг 3.18 – Створення таблиці «ReportPeriod»

```
create table ReportPeriod
(
ID_R_Period int not null primary key identity (1,1),
Period_Name varchar (200),
StartDate datetime,
EndDate datetime,
[Description] varchar (max)
)
```

Таблиця 3.12 – «ReportsType»

Атрибут	Тип	Примітка
ID_RType	int	PK
RType_Name	varchar(200)	
Description	varchar(MAX)	

Таблиця 3.13 – «ReportPeriod»

Атрибут	Тип	Примітка
ID_R_Period	int	PK
Period_Name	varchar(200)	
StartDate	datetime	
EndDate	datetime	
Description	varchar(MAX)	

Детальна інформація про показники звіту зберігається у таблиці «ReportDetail», код створення якої наведено у лістингу 3.18, а схему – у вигляді табл. 3.14.

Лістинг 3.18 – SQL-запит для створення таблиці «ReportDetail»

```
create table ReportDetails
(
ID_ReportDetails int not null primary key identity (1,1),
ID_Person int foreign key references Patient (ID_Patient),
ID_Report int foreign key references Report (ID_Report),
ID_SGroup int foreign key references StatisticGroup(ID_SGroup),
ID_Parent_test int foreign key references Test(ID_Test),
PeroidValue decimal,
[Description] varchar (MAX)
)
```

Таблиця 3.14 – Схема таблиці «ReportDetail»

Атрибут	Тип	Примітка
ID_ReportDetails	int	PK
ID_Person	int	FK
ID_Report	int	FK
ID_SGroup	int	FK
ID_Parent_test	int	FK
PeroidValue	decimal(18, 0)	
Description	varchar(MAX)	

Оскільки, однією із задач розподіленої комп'ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 є формування різного роду статистичних даних, то для зберігання таких даних передбачено таблицю «StatisticGroup».

SQL-запит створення «StatisticGroup» показано у лістингу 3.19, а схема створеної таблиці наведено у вигляді табл. 3.15.

Лістинг 3.19 – SQL-запит на створення таблиці «StatisticGroup»

```
create table StatisticGroup
(
ID_SGroup int not null primary key identity (1,1),
SGroupName varchar (max),
ID_Parent_SGroup int foreign key references
StatisticGroup(ID_SGroup)
)
```

Таблиця 3.15 – «StatisticGroup»

Атрибут	Тип	Примітка
ID_SGroup	int	PK
SGroupName	varchar(MAX)	
ID_Parent_SGroup	int	FK

Реалізувавши сутності предметної області засобами мови SQL, у середовищі MS SQL Server Management Studio 2019 Express згенеровано ER-діаграму бази даних, що показана на рис. 3.2. Дана відображає зв'язки між сутностями, представленими у вигляді реляційних відношень.

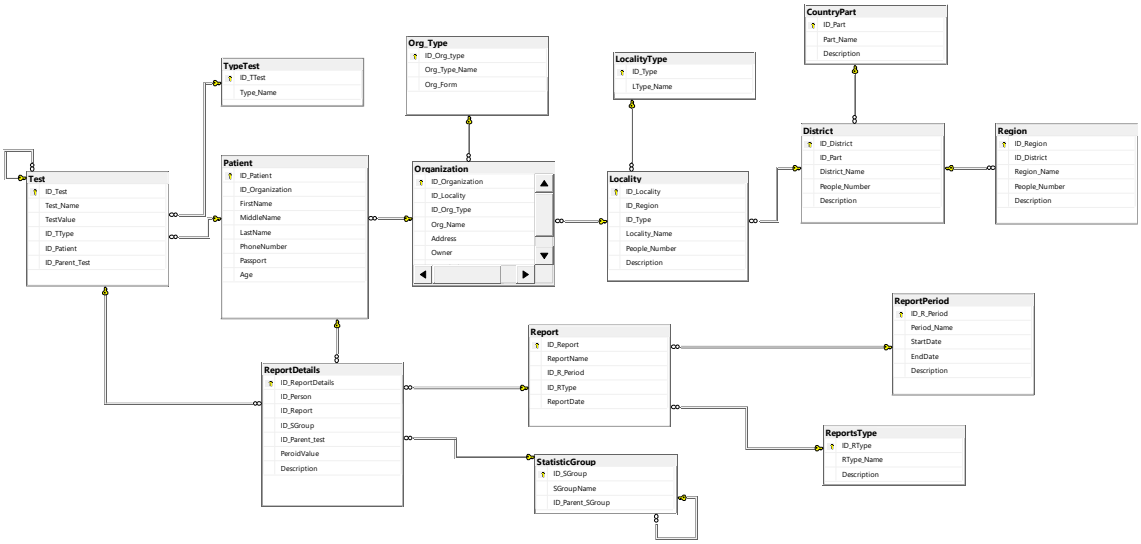

Рисунок 3.2 – ER-діаграма бази даних збору та аналізу інформації щодо захворюваності на COVID-19

Створена база даних є нормалізованою і відповідає третій нормальній формі.

3.3 Реалізація користувацького інтерфейсу і тестування програмного забезпечення комп’ютерної системи

Інтерфейс користувача прикладного клієнтського додатку розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 розроблено із врахуванням вимог до програмного засобу та специфіки прикладних задач, в яких буде застосовуватись система.

Стартовий екран програми представляє собою форму, яка відображає список доступних для роботи довідників та надає засоби роботи з даними. З допомогою кнопок стартової форми можна здійснювати створення, редагування та видалення довідників, експорт довідника, перегляд та редагування записів, пошук в базах даних, налаштування (рис. 3.3).

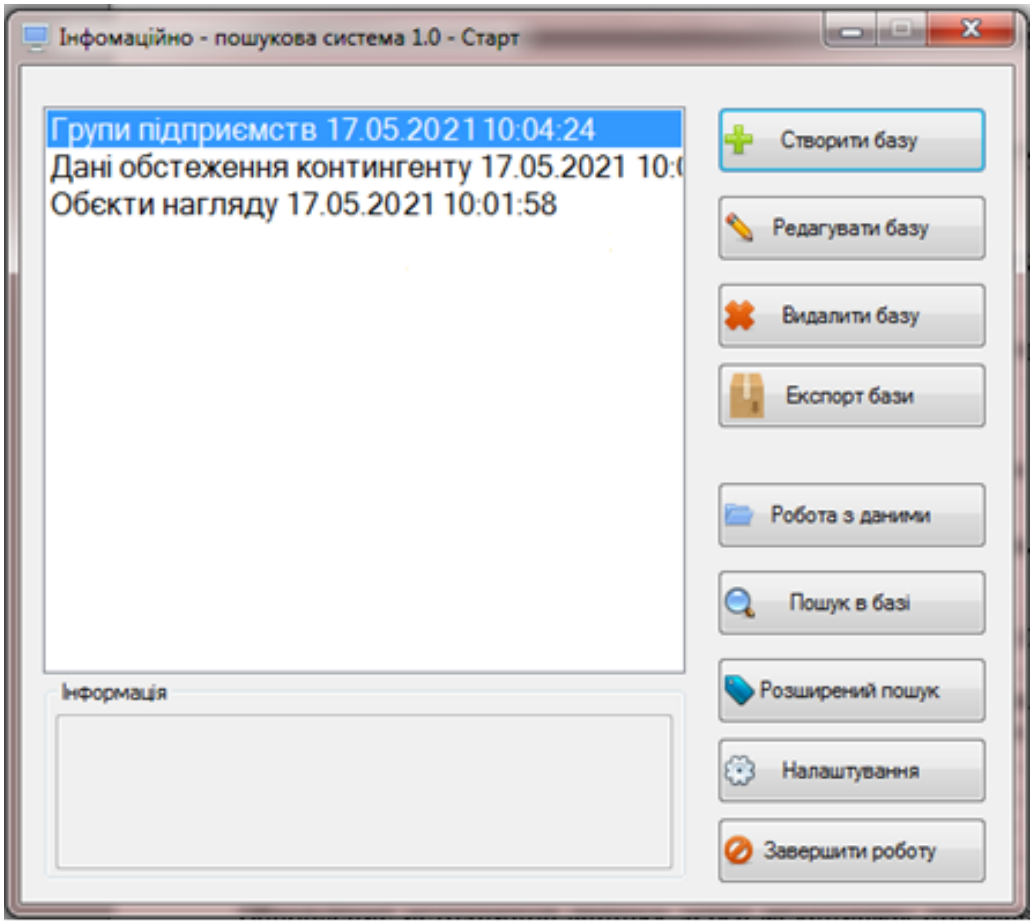

Рисунок 3.3 – Вигляд стартового вікна програми

За допомогою системи підтримки звітності користувачі можуть створювати та редагувати спеціалізовані довідники для зберігання статистичної інформації.

Форма створення та редагування схеми довідників включає необхідні поля для встановлення значень атрибутів довідника. Крім того зручне поєднання стилю елементу відображення – таблиці та форми введення даних дозволяють виконувати внесення та редагування полів каталогу (рис. 3.4).

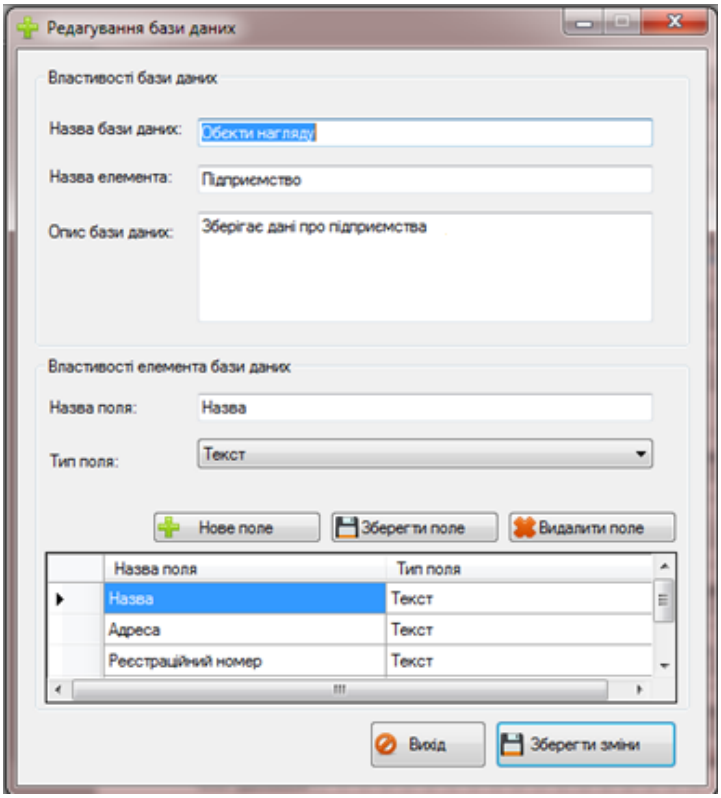

Рисунок 3.4 – Редагування схеми довідника «Об'єкти нагляду (Організації)»

Для внесення потрібної звітної інформації система надає інтерфейс перегляду та внесення даних. Інтерфейс виконано з врахуванням вимог до програмного засобу та надає прості у використанні, компактні та продуктивні елементи роботи з даними. Використано розширені елементи роботи з даними – таблиці, навігатори, форму деталей, панель інструментів. Панель інструментів інтегрована із навігатором містить кнопки-піктограми для швидкого виклику найбільш часто використовуваних операцій – додавання, видалення та збереження даних (рис. 3.5).

Окрім цього, на даній формі існує можливість:

– створення нового запису;
– видалення існуючих записів;
– збереження змін в існуючі записи;
– переміщення даних у зручному для користувача порядку.

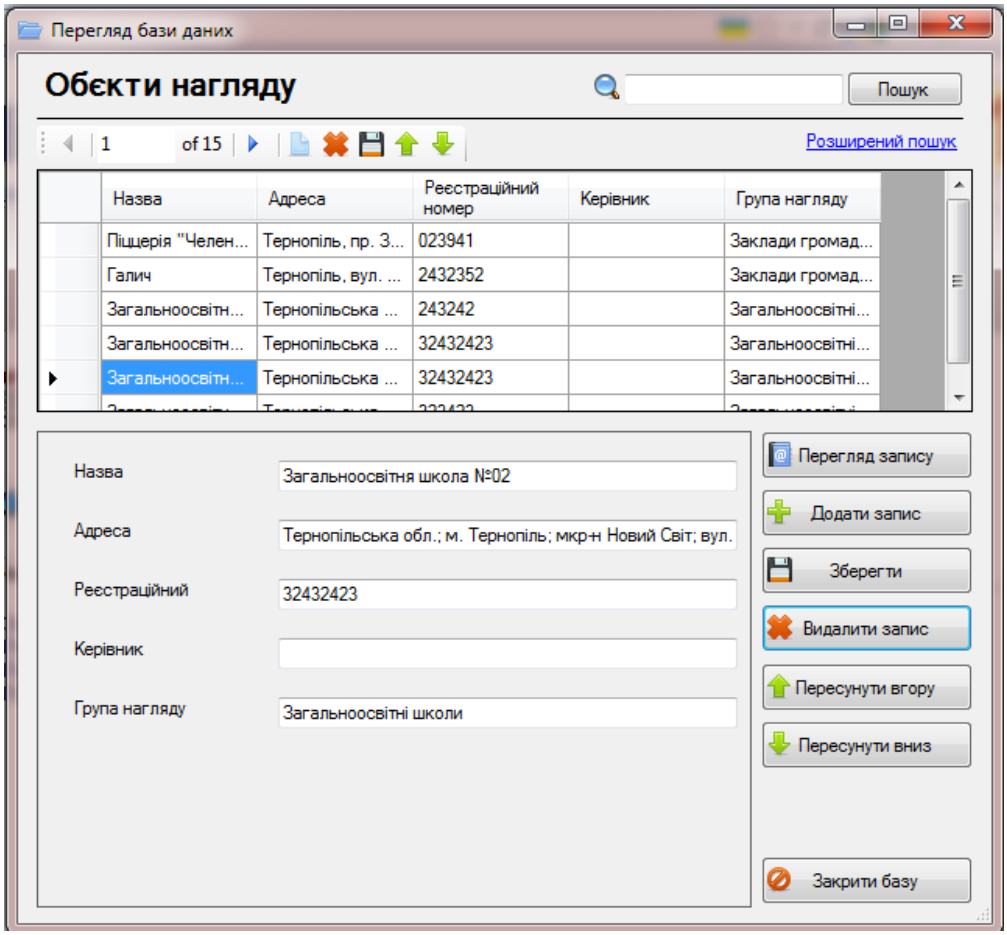

Рисунок 3.5 – Вікно внесення та редагування даних довідника підприємств

За допомогою кнопки «Перегляд даних» (рис. 3.6) користувач може викликати форму для перегляду чи редагування даних конкретного запису (наприклад, по підприємству, де зафіксовано захворюваність на COVID-19).

Візуальні елементи форми розроблені та налаштовані таким чином, щоб користувач міг безперешкодно виконувати масштабування вікна та міг переглядати вміст будь-якого поля.

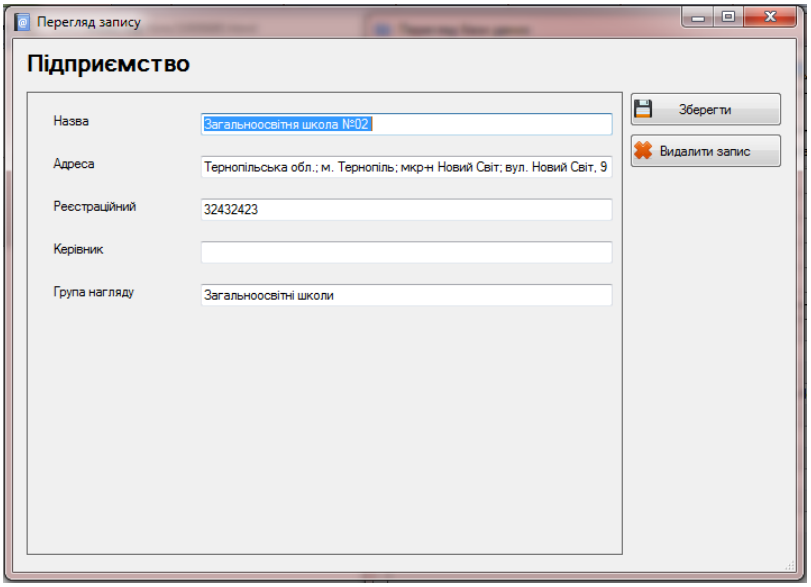

Рисунок 3.6– Вікно перегляду та редагування даних конкретного підприємства

Прикладний клієнтський додаток для роботи із даними обласного лабораторного центру надає також гнучкі засоби для пошуку інформації, як наведено на рис. 3.7.

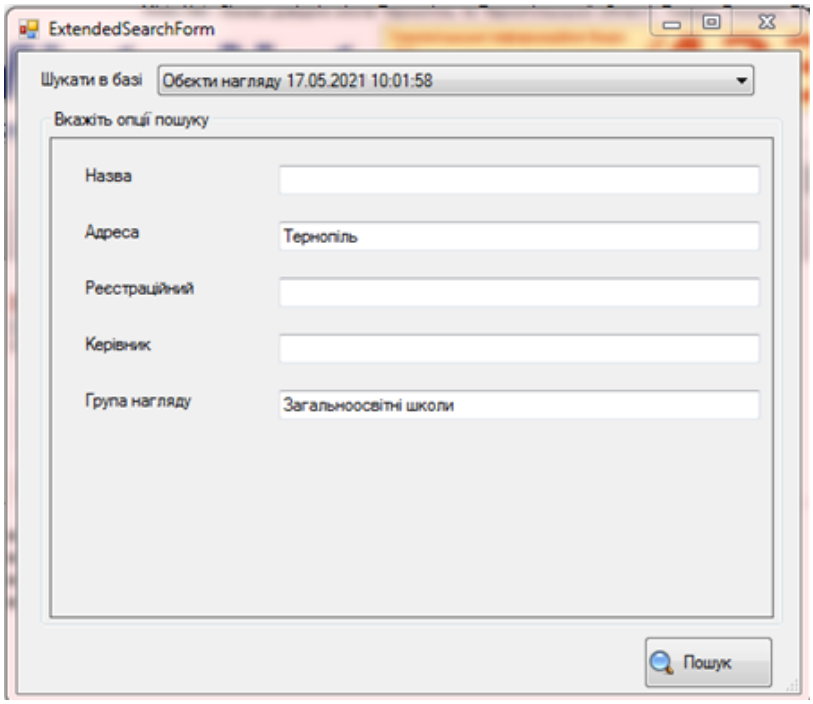

Рисунок 3.7 – Вигляд форми розширеного пошуку

Результати пошуку представляються у вигляді таблиці та форми детальних записів із можливістю перегляду, редагування та видалення певних значень і

показано на рис. 3.8. Крім того, форма відображення результатів пошуку дозволяє перейти до режиму перегляду всього довідника даних або повторно застосувати пошук.

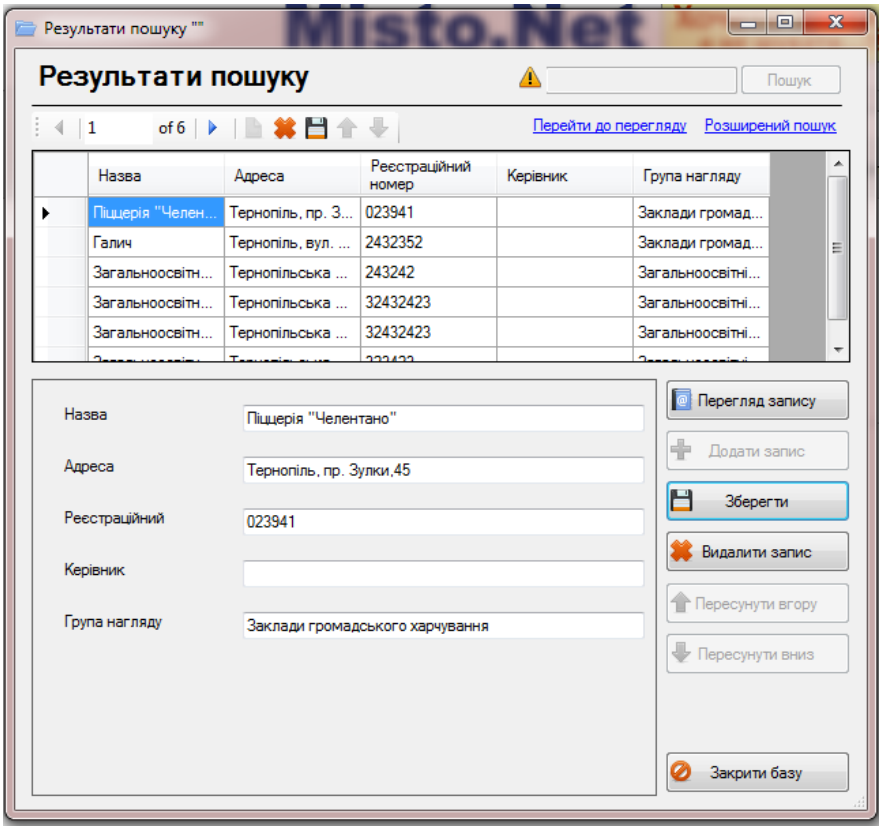

Рисунок 3.8 – Вивід результатів пошуку

Крім розширеного та налаштовуваного пошуку система надає можливість простого пошуку на співпадіння значень. Вікно простого пошуку доступне з форми перегляду даних довідника або з головної форми (рис. 3.9).

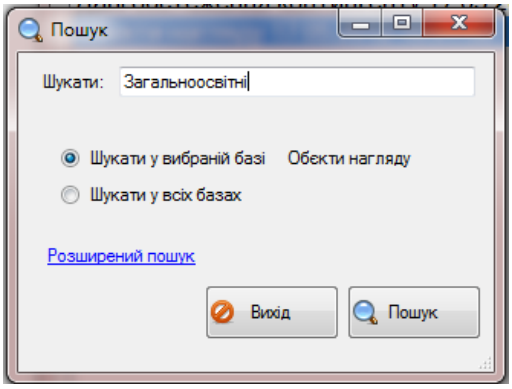

Рисунок 3.9 – Вікно простого пошуку

Вікно простого пошуку надає можливість знаходити інформацію по усіх базах-довідниках інформації. Вивід знайдених відповідників виводиться на форму в табличному вигляді та сортуванням залежно від джерела даних, як показано на рис. 3.10.

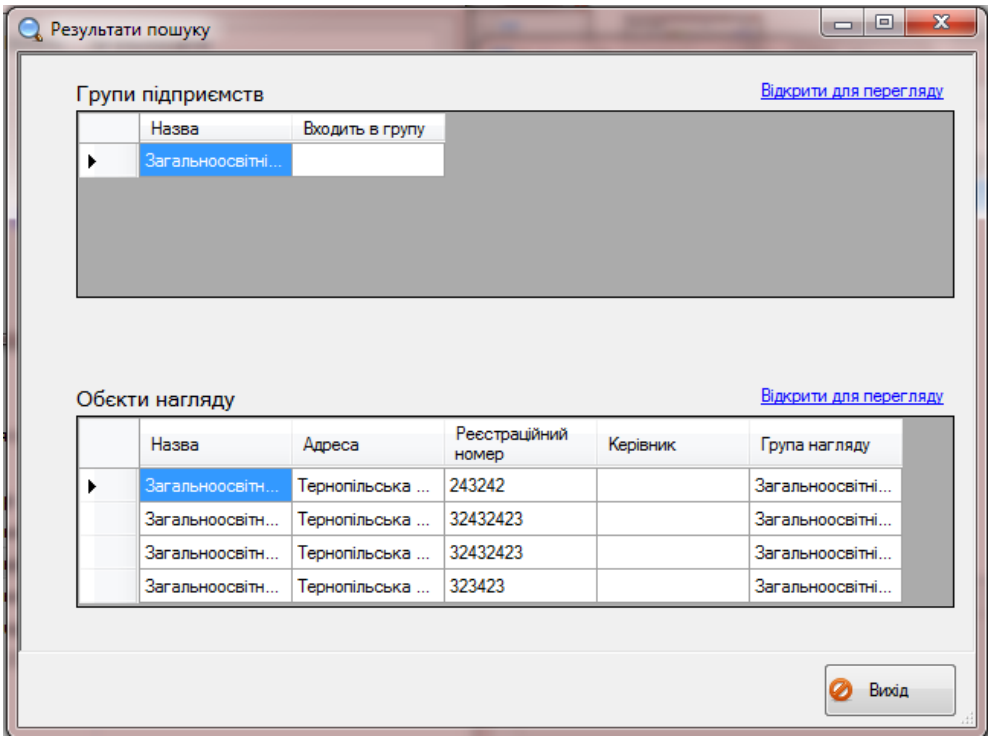

Рисунок 3.10 – Вивід результатів розширеного пошуку

В результаті обробки статистичної інформації в довідниках та базах знань системи користувач може згенерувати звіт визначеної форми використовуючи засоби експорту інформації. Згенеровані звіти представляють собою документи електронних таблиць офісного пакета Excel.

Представлений інтерфейс користувача в системі збору та аналізу даних щодо захворюваності на COVID-19 є графічною оболонкою, що базується на використанні сучасної віконної парадигми.

Екрани прикладного користувацького додатку організовують роботу лікарів-епідеміологів з метою покращення процесу обробки статистичної інформації та підвищення ефективності роботи організації.

3.4 Розгортання програмного забезпечення на локальних вузлах розподіленої комп’ютерної системи

Для зручності розповсюдження програмних пакетів та розгортання в цільових клієнтських системах середовище розробки Visual Studio пропонує ряд сценаріїв інсталяції програмних додатків. Запропоновані інструменти, що входять до складу середовища розробки мають на меті стандартизацію розгортання систем у поєднанні із одночасною гнучкістю сценаріїв установки.

Кожен з цих варіантів розгортання володіє власними перевагами та недоліками використання в певних сценаріях інсталяції. Стандартним рекомендованим способом розгортання клієнтських настільних додатків є проект інсталяційного пакету (Setup Project), що дозволяє виконувати розгортання прикладного додатку з допомогою покрокового діалогу.

Для початку інсталяції прикладного користувацького додатку необхідно виконати запуск пакету Microsoft Installer (файл із розширенням .msi). На екрані з’явиться початкове вікно інсталятора з пропозицією розпочати процес встановлення клієнтського додатку (рис. 3.11).

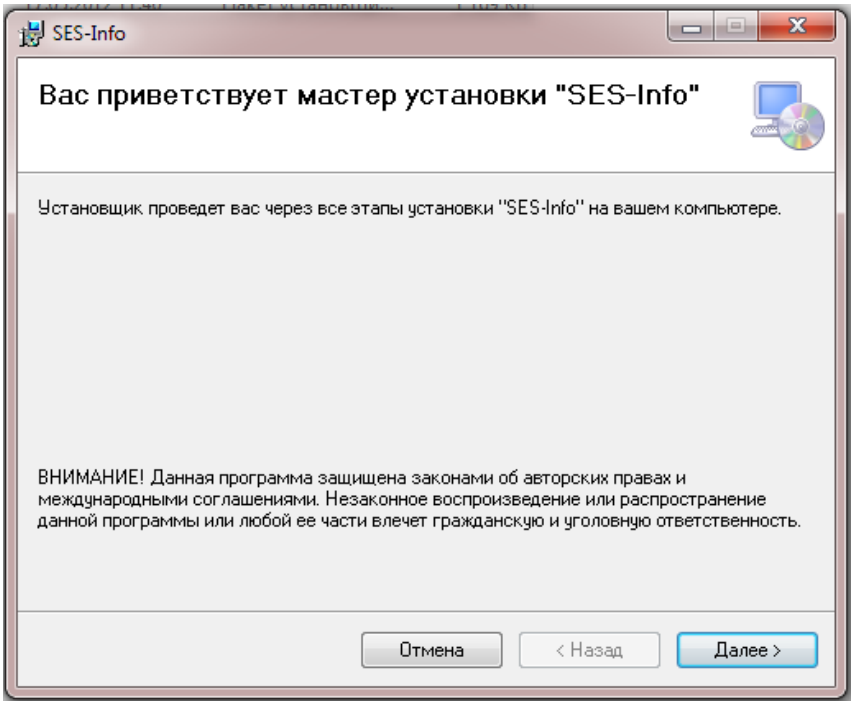

Рисунок 3.11 – Початкове вікно діалогу майстра інсталяції

При натисканні кнопки «Далі» наступним кроком діалогу буде пропозиція вибрати розташування в файловій системі куди буде встановлюватись програма та опції доступності програми для запуску (рис. 3.12).

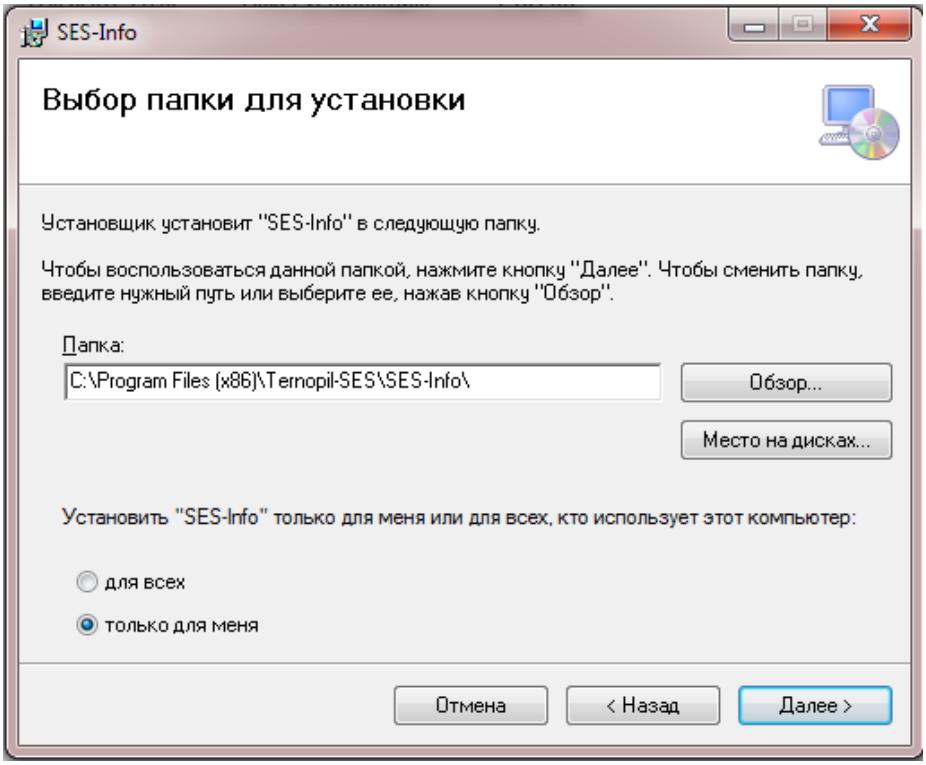

Рисунок 3.12 – Вибір місця розташування в файловій системі для встановлення програми

Після вибору необхідного місця розташування та опцій доступу до встановленої програми можна перейти до наступного кроку встановлення з допомогою кнопки «Далі».

Наступне діалогове вікно (рис. 3.13) інформує про готовність системи до інсталяції та попереджує про те що далі майстер встановлення розпочне інсталяцію програми і що на даний момент ще є можливість змінити опції інсталяції. У випадку якщо в системі відсутні будь-які компоненти залежності для програмного пакету їх буде встановлено.

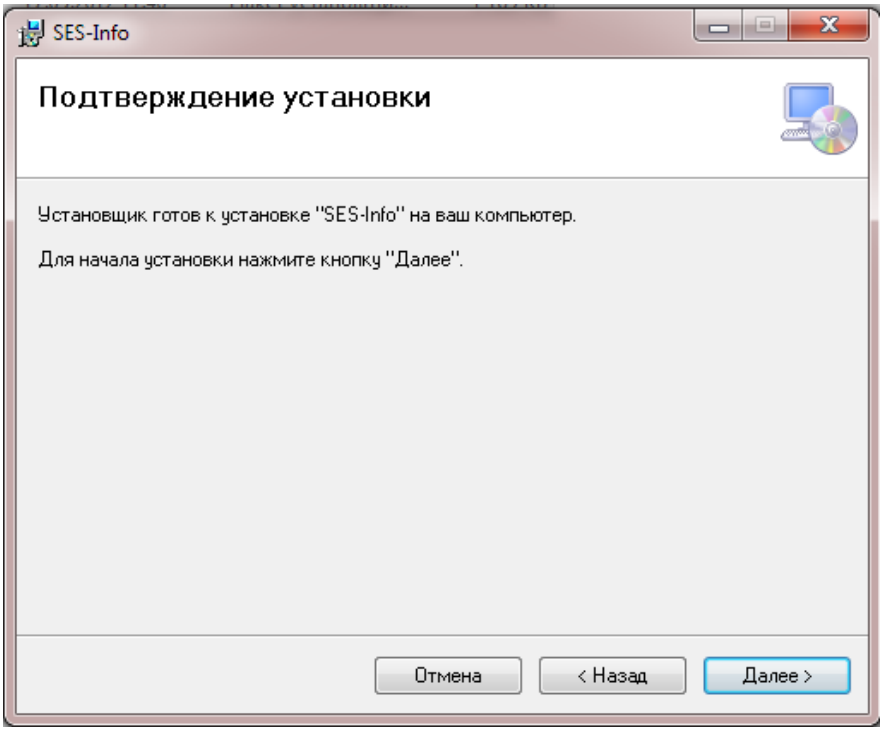

Рисунок 3.13 – Екран погодження для початку інсталяції

Для початку процесу копіювання файлів у файлову систему та розгортання необхідних компонентів необхідно натиснути кнопку «Далі». Після цього з’явиться вікно, яке показано на рис. 3.14.

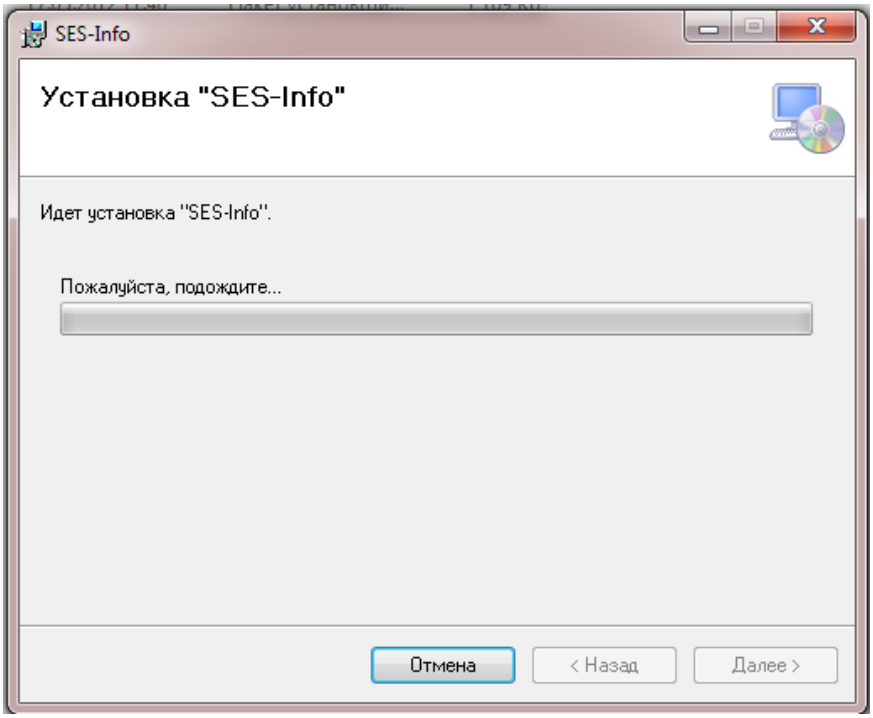

Рисунок 3.14 – Запуск процесу розгортання клієнтського додатку

Майстер інсталяції розпочне автоматичне копіювання файлів, реєстрацію та розгортання компонентів, це може зайняти деякий час. Прогрес інсталяції файлів буде відображатись із допомогою стрічки прогресу установки (рис. 3.15).

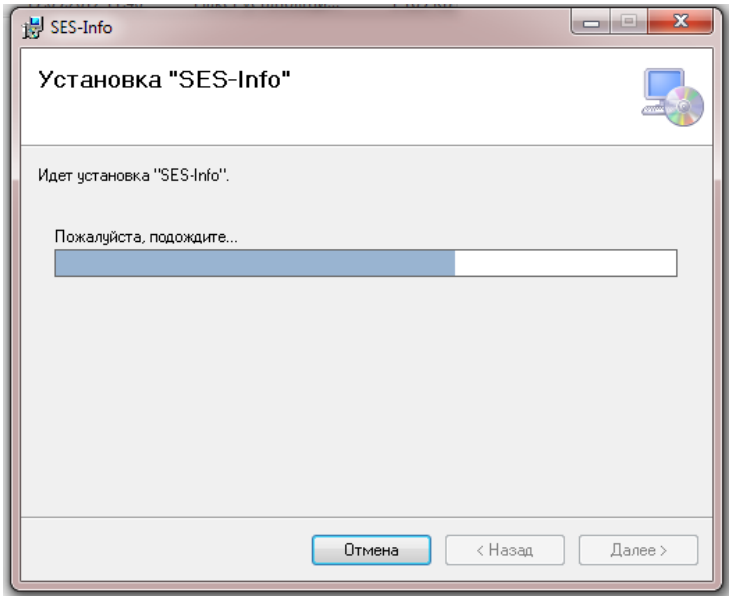

Рисунок 3.15 – Процес копіювання та реєстрації компонентів програми

Про вдале завершення установки програмних компонентів буде повідомлено з допомогою спеціального вікна діалогу. Процес установки можна завершити натиснувши кнопку «Вихід».

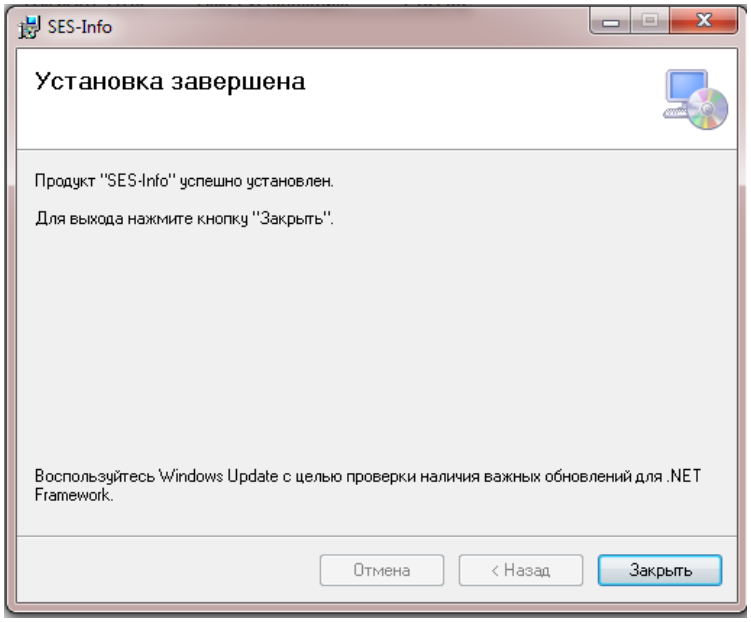

Рисунок 3.16 – Вдале завершення інсталяції

Після завершення процесу інсталяції програмне забезпечення розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності COVID-19 готове до використання. Запуск програми виконується за допомогою виконуваного файлу програми за розташуванням, що було вказане в процесі встановлення програми або з допомогою ярлика на робочому столі користувача.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Фактори трудової діяльності та умови праці користувачів ПК

Дослідження, проведені фахівцями Всесвітньої організації охорони здоров'я (ВООЗ) показали, що у професійних операторів та канцелярських службовців, які у своїй діяльності використовують ПК, частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Необхідно зазначити, що вже в перші роки впровадження ВДТ в Європі та США була зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів. Однак, в той час основна увага приділялась розвитку техніки, а людина залишалась без необхідного захисту.

В умовах сучасного виробництва, яке характеризується масовим характером та широким застосуванням комп'ютерної техніки попередні пріоритети зазнали суттєвої трасформації. У центрі уваги вітчизняних та зарубіжних фахівців є питання щодо визначення характеру та умов праці користувачів комп'ютерів, функціональних змін у динаміці виконання трудових завдань, захворюваності та стану здоров'я, розробки засобів захисту.

Дослідження медиків-гігієністів, психологів, світлотехніків та фахівців з охорони праці та ергономіки показали, що сучасна професія користувача ПК належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних висококоординованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервово-емоційним компонентом,

особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищих психічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання.

Встановлено, що стан організму користувача значно залежить від типу роботи з ПК та умов її виконання. В загальному усі користувачі комп'ютерів поділяються на професіоналів та непрофесіоналів. До останніх можна віднести осіб, які використовують комп'ютер епізодично і він є для них не основним, а тільки допоміжним засобом (науково-технічні працівники, бібліотекарі, студенти, школярі, торгівельні працівники).

Діяльність професіоналів можна поділити на три групи:

– діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового напруження (робота операторів комп'ютерного набору, працівників довідкових служб);

– діяльність, яка пов'язана із здійсненням логічних операцій, що постійно повторюються – це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва;

– діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму (робота інженера-програміста, диспетчерів руху залізничного транспорту, аеропортів).

Необхідно зазначити, що такий поділ досить умовний, оскільки дане питання ще не достатньо розроблене і потребує детального вивчення. Проте, зрозуміло, що для кожної категорії користувачів комп'ютерів характерні свої особливості впливу комплексу несприятливих факторів трудового процесу та умов праці.

В Інституті медицини праці Академії медичних наук України проводились дослідження інтенсивності захворюваності осіб, що використовують у своїй роботі комп'ютер. Була вивчена захворюваність працівників з різною тривалістю використання комп'ютерів та характером діяльності самих користувачів. Розглядалися три групи користувачів: у першу

увійшли системні інженери-програмісти (тривалість роботи за комп'ютером більше 6 год. на день), у другу — інженери-економісти, які у своїй роботі використовують уже розроблене програмне забезпечення (тривалість роботи від 4 до 6 год.), у третю — математики-постановники завдань, які використовували комп'ютери не більше, ніж 2 год. на день. Дані про захворюваність різних груп користувачів комп'ютерів та контрольної групи наведено у таблиці 4.1.

Таблиця 4.1 – Рівень захворюваності (%) осіб

Стан здоров'я	Користувачі ВДТ			Контрольна група
	1 група	2 група	3 група	
Функціональні порушення ЦНС (астенопічний синдром та ін.)	15,6	8,2	6,3	2,7
Хвороби системи кровообігу	57,7	60,3	29,2	23,0
Хвороби органів дихання	20,0	21,7	11,2	4,1
Хвороби органів травлення	40,0	38,6	29,8	18,9
Здорові	6,7	20,1	29,8	46,6

Так, здорових серед обстежених користувачів ВДТ виявилось у кілька разів менше, ніж у контрольній групі. З наведених даних видно, що фізіологічні порушення частіше спостерігаються у користувачів, які довше та інтенсивніше використовували ВДТ.

Крім того, за даними ряду авторів у користувачів, які інтенсивно використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість) [15]. Працівниками кафедри охорони праці та екології Української академії друкарства та Українського науково-дослідного інституту поліграфічної промисловості ім. Т. Шевченка проведені дослідження умов праці та особливостей трудового процесу на комп'ютеризованих робочих місцях низки

підприємств, що займаються видавничо-поліграфічною діяльністю (редакції, видавництва, друкарні). Було, зокрема, встановлено, що за суб'єктивними показниками (скарги) робота з ВДТ викликає різноманітні симптоми негативного впливу на здоров'я користувачів. У таблицях в [15] наведені характеристики скарг операторів комп'ютерного набору та редакторів і коректорів, які в процесі своєї роботи використовували комп'ютер. Результати досліджень наглядно ілюструють, що у працівників різних професійних груп, що працюють за відеотерміналом комп'ютера переважають "очні" симптоми. Часті також головний біль та загальна втома, особливо в кінці робочого дня. Причому у коректорів та редакторів такі симптоми зустрічаються частіше. Досить значний відсоток скарг пов'язаний з опорно-м'язовою системою (біль в області спини та шиї, втома м'язів рук), особливо в операторів комп'ютерного набору. Характерним також є той факт, що чим більший стаж роботи за комп'ютером, тим очевидніший його несприятливий вплив на здоров'я користувача.

4.2 Надзвичайні ситуації, викликані вибухами і способи захисту від них

Вибух – це вивільнення значної кількості енергії в обмеженому об'ємі за короткий проміжок часу. Вибух здійснюється частіше за все за рахунок вивільнення хімічної енергії вибухових речовин [15].

Вибухові речовини (ВР) – це порох, динаміт, тротил, нітрогліцерин та інші хімічні речовини, їх сполуки або суміші, здатні вибухнути без доступу кисню.

За складом ВР поділяються на вибухові хімічні сполуки і вибухові суміші, а за призначенням – на ініціювальні (первинні) та бризантні (вторинні). Ініціювальна ВР (азид свинцю, гримуча ртуть, тетразен тощо) - високочутлива до найпростіших імпульсів маса (удар, тертя, наколювання, електрична іскра тощо), яка застосовується для збудження вибухових перетворень у зарядах вторинних ВР. Бризантні ВР мають велику швидкість детонації, тобто швидкість вибухового перетворення - до 8 км/сек; серед них можна виділити гексоген, октоген, тротил, тетрил [16].

Під вибуховими пристроями (ВП) розуміють саморобні чи виготовлені промисловим способом вироби одноразового застосування, спеціально підготовлені і за певних обставин спроможні за допомогою використання хімічної, теплової, електричної енергії або фізичного впливу (вибуху, удару) створити вражаючі ефекти - спричинити смерть, тілесні ушкодження, істотну матеріальну шкоду шляхом вивільнення, розсіювання або впливу ударної хвилі, спалаху світла, теплової енергії, осколків, токсичних речовин, радіоактивних матеріалів, біологічних агентів. ВП можуть бути штатні та виготовлені з використанням побутових предметів – портфелів, іграшок, пакетів, електроприладів. Вони можуть спрацьовувати від натискування, падіння, піднімання, впливу радіохвиль певної частоти [16].

Основними елементами ВП є заряд ВР або вибухонебезпечної суміші, засоби ініціювання (підривник, детонатор), обладнання для приведення ВП у дію та корпус ВП, хоч останній може бути й безоболонковим.

Підрив вибухового пристрою може здійснюватись механічним, електричним або вогневим способами. Для першого використовують механічний підривник – детонатор, ударник, пружину та запобіжну чеку. Для другого способу підриву застосовують електродетонатори, електробатареї або інше джерело живлення: ініціювання підриву відбувається шляхом замкнення електромережі і підриву детонатора (дистанційно або за допомогою проводів). Третій спосіб підриву передбачає для ініціювання детонатора використання вогнепровідного шнура.

Вибух – це процес надзвичайно швидкого горіння, що супроводжується швидким зростанням тиску і має велику руйнівну силу.

Якщо у повітряному середовищі виникає така концентрація пилу, парів або газів, яка досягає межі вибуху, то при наявності відкритого джерела вогню відбудеться вибух (табл.4.2).

Пил деяких речовин має такі значення нижньої межі вибуху (г/м}): цукор - 8,9; торф - 10,1; сіно - 20,2; тирса - 65,0.

Найбільш небезпечним є високодисперсний пил, бо він має велику сумарну поверхню, що створює підвищену хімічну активність.

Вибухи та пожежі можуть виникати за таких обставин:

– у початковий період експлуатації виробництва — період притирання елементів технологічного обладнання (недоліки допущенні у процесі проектування, неякісне виконання монтажних робіт);

– в основний період експлуатації виробництва (через несправність контрольно-вимірювальних приладів та елементів обладнання, порушення вимог безпеки, недостатній нагляд і контроль, незадовільні планово-профілактичні ремонти);

– у період так званого "старіння " елементів технологічного обладнання (через корозію матеріалів, зношеність деталей, відсутність капітальних і поточних ремонтів).

Таблиця 4.2. – Концентраційні межі вибуху (займання) деяких горючих газів і парів, легкозаймистих і горючих рідин

Речовина	Нижня межа		Верхня межа	
	% за об'ємом	г/м^3 при 20^{0}С	% за об'ємом	г/м^3 при 20^{0}С
Метиловий спирт	6,7	46,5	38,5	512,0
Етиловий спирт	3,61	50,0	19,0	363,0
Бутан	1,8	37,4	8,5	204,8
Метан	5,28	16,66	15,4	102,6
Пропан	2,31	36,6	9,5	173,8
Ацетилен	2,50	16,5	82,0	885,6
Пропілен	2,30	34,8	11,1	169,0
Етилен	3,11	35,0	35,0	406,0
Бензол	1,43	42,0	9,5	308,0
Ксилол	1,0	44,0	7,6	34,0
Толуол	1,25	38,2	7,0	268,0
Ацетон	2,91	38,6	13,0	314,0
Аміак	17,0	112,0	27,0	189,0
Сірководень	4,0	61,0	44,5	628,0
Бензин паливний	2,4	137,0	4,9	281,0
Бензин розчинник	1,9	-	5,1	-
Водень	4,09	3,4	80,0	66,4

Причинами виникнення пожеж можуть бути конструктивні недоліки та порушення експлуатації пічного опалення, електрообладнання та

електроустаткування, інженерних комунікацій, дефекти обладнання; організаційні порушення режимів технологічних процесів, похибки при виконанні технологічних процесів, необережні дії персоналу та низький рівень їх кваліфікації, порушення правил при поводженні з вогнем.

Всі різноманітні причини пожеж можна об'єднати у дві великі групи, що пов'язані:

– I група - з недопустимою, з точки зору пожежної безпеки, появою горючого середовища в умовах, де є джерело вогню (розрив трубопроводів в котельнях, що працюють на рідкому паливі, підтікання паливних ліній, аварійний стан обладнання, викиди бітуму при варінні);

– II група - з недопустимою появою джерел вогню при наявності горючої суміші і окисника.

Сюди відносяться пожежі обумовлені такими чинниками як:

– порушення режимів роботи технологічного та інженерно-технічного обладнання;

– недоліки монтажу та порушення експлуатації електрообладнання, електропроводки, електроапаратури (коротке замикання, іскріння, перенавантаження проводів);

– недоліки при облаштуванні та експлуатації опалювальних систем та установок;

– недопустиме підвищенням температури при адіабатичному стисненні (робота компресорного устаткування);

– перегрів речовин, що обробляються, коли їх температура досягає температури самозаймання;

– порушення режиму зберігання та обробки самозаймистих речовин і матеріалів;

– необережне поводження з вогнем, незнання правил та норм пожежної безпеки, недбале ставлення до своїх обов'язків, навмисний підпал, накопичення електростатичних розрядів.

Безпосередньо причиною пожежі може стати у непередбачений час, у непередбаченому місці, з точки зору пожежної безпеки, поява того чи іншого

компоненту, який бере участь в процесі горіння, а відтак, з метою профілактики, недопустимими є такі умови, що можуть призводити до неконтрольованих процесів горіння.

Основними вражаючими факторами вибухів є:

– повітряна ударна хвиля (ПУХ), що виникає при ядерних вибухах, вибухах детонуючих та ініціюючих речовин, при вибухових перетвореннях хмар паливно-повітряних сумішей, вибухів резервуарів з перегрітої рідиною і резервуарів під тиском;

– осколкові поля, створювані уламками різного роду об'єктів.

Основними параметрами вражаючих факторів є:

– повітряна ударна хвиля – надлишковий тиск в її фронті;

– осколкове поле – кількість осколків, їх кінетична енергія і радіус розльоту.

У результаті дії вражаючих факторів вибуху відбувається руйнування або пошкодження будівель, споруд, обладнання, елементів комунікації, і загибель людей і тварин.

Вторинними наслідками вибухів є ураження об’єктів, які знаходяться всередині приміщень, уламками завалених конструкцій будівлі. У результаті вибухів можуть виникнути пожежі, витік небезпечних речовин з пошкодженого обладнання. При пожежах і вибухах люди отримують термічні і механічні травми. Характерні опіки верхніх дихальних шляхів, тіла, черепно-мозкові травми, множинні переломи і удари, комбіновані ушкодження.

Для запобігання пожеж і вибухів необхідно виключити можливість утворення горючого і вибухонебезпечного середовища, а також запобігти появі в цих середовищах джерел загоряння.

Уражаюча дія вибуху, незалежно від його природи, може призводити до значних матеріальних і людських втрат. Особливо актуальним є питання збереження життя людей і забезпечення функціонування підприємств стратегічного значення в Україні в умовах війни на сході.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра спроектовано архітектуру розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19, розроблено програмне забезпечення для підтримки відповідних бізнес-процесів та формування статистичної звітності за обраний період часу.

У системі реалізовано можливість заповнення довідників щодо осіб, які проходять тестування на COVID-19, регіонів та населених пунктів, підприємств на яких працюють особи та ряду інших. Усі довідники у системі категоризовані, що забезпечує зручність та швидкість пошуку інформації.

Окрім того, на локальних вузлах розподіленої комп’ютерної системи, розгорнуто програмне забезпечення для забезпечення можливості реєстрації пацієнтів, фіксації результатів тестування, формування звітів і статистичних даних в розрізі різних критеріїв: дати і часу, віку осіб і т.п.

У першому розділі кваліфікаційної роботи проведено аналіз технічного завдання, проаналізовано та обгрунтовано доцільність застосування механізмів змішаної фрагментації і реплікації реляційних відношень при організації розподіленої системи.

У другому розділі роботи спроектовано архітектуру розподіленої комп’ютерної системи, визначено функції та ролі користувачів системи, проаналізовано предметну область, у результаті чого одержано набір сутностей та відповідних атрибутів. Окрім цього визначено інформаційні потоки, які циркулюють у розподіленій комп’ютерній системі.

У третьому розділі, із застосуванням мови програмування C# і технології Windows Forms, реалізовано прикладний програмний додаток для збору та аналізу даних щодо захворюваності на COVID-19. Розроблено алгоритми роботи програми та опрацювання даних окремими модулями.

Базу даних розподіленої комп’ютерної системи збору та аналізу даних щодо захворюваності на COVID-19 реалізовано на основі реляційного підходу із застосування MS SQL Server 2019 Express. Проведено її нормалізацію, у результаті чого база даних перебуває у третій нормальній формі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лаврищева Е.М. Основы технологической подготовки разработки прикладных программ систем обработки данных. АН УССР, Ин-т кибернетики им. В.М. Глушкова; 87-5. Киев. 1987. 30 с.

2. Андон Ф.И., Лаврищева Е.М. Методы инженерии распределенных компьютерных систем. К.: Наукова думка. 1997. 228 с.

3. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. К.: Наукова думка. 1991. 213 с.

4. Лаврищева Е.М. Парадигма интеграции в программной инженерии. Проблемы программирования. 2000. № 1-2. С. 351 – 360.

5. Нейман В.И. Структуры систем распределения информации. - 2-е изд, перераб. и доп. М.: Радио и связь. 1983. 216 с.

6. Виейра Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс. .: Пер. с англ. М.: ООО «И.Д. Вильямс». 2009. 816 с.

7. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К. C# 2008 и платформа .NET 3.5 для профессионалов. Пер. с англ. М. : ООО "И.Д. Вильямс". 2009 р. 1392 с.

8. Макки А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. СПб «Вильямс». 2010. 446 с.

9. Пасічник В., Резніченко В. Організація баз даних та знань. К:. Видавнича група BHV, 2006. 384 с.

10. Бьорнс Б. Распределенные системы. Паттерны проектирования. Питер. 2019. 224 с.

11. Карпов Ю.Г. Model checking. Верификация параллельных и распределенных программных систем. БХВ-Петербург. 2010. 560 с.

12. Чекмарев Ю. Вычислительные системы, сети и телекоммуникации. М:ДМК Пресс. 2009. 184 с.

13. Malaiya Y.K., Denton J. What do the Software Reliability Growth Model Parameters Represent. Proc. IEEE-CS Int. Symp. on Software Reliability Engineering ISSRE. Nov. 1997. P. 124 - 135.

14. Chulani S. Constructive quality modeling for defect density prediction: COQUALMO. International Symposium on Software Reliability Engineering (ISSRE'99), Boca Raton, November 1-4. 1999. P. 105-114.

15. Жидецький В.Ц. Охорона праці користувачів комп’ютерів. Львів: Афіша, 2000. 176 с.

16. Желібо Є., Заверуха Н., Зацарний В. Безпека життєдіяльності. К.: 2001. 483 с.

Додаток А.

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“___” _____2021 р

РОЗПОДІЛЕНА КОМП’ЮТЕРНА СИСТЕМА ЗБОРУ ТА АНАЛІЗУ ДАНИХ ЩОДО ЗАХВОРЮВАНОСТІ НА COVID-19

ТЕХНІЧНЕ ЗАВДАННЯ

на _9_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»

Керівник кваліфікаційної роботи

_____ к.т.н., доц. Луцків А.М.

«____» _____ 2021 р.

«ВИКОНАВЕЦЬ»

Студент групи СІс-44

_____ Бєляєв Ю.В.

«____» _____ 2021 р.

Тернопіль 2021

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.159.00.00

1.2 Виконавець

Студент групи СІс-44, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Бєляєв Юрій Віталійович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№ 4.7-97 від 10.02.2021 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 10.02.2021 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2021 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень з допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19 призначена для автоматизації процесу збору, накопичення, опрацювання і централізованого керування статистичними показниками щодо захворюваності пандемічним вірусом COVID-19. Дана система може бути використана обласними лабораторними центрами, утвореними на базі санітарно-епідеміологічних служб, та повинна забезпечувати щоденний моніторинг стану захворюваності в окремих регіонах. Окрім цього, розподілена комп’ютерна система повинна володіти функціональністю щодо формування та надсилання звітів у централізоване сховище і надавати авторизований доступ передбаченим типам користувачів. Основна функціональність системи збору та аналізу даних передбачає зручне опрацювання статистичної інформації, візуалізацію та побудову відповідних видів діаграм і графіків у відповідності до показників, виводити кількість нових випадків захворюваності за останній звітний період установ, забезпечувати централізоване зберігання даних, модифікацію інформації користувачами в межах їх компетентності.

2.2 Мета створення системи

Метою створення системи є автоматизація процесу збору та опрацювання даних щодо поширення захворюваності на COVID-19 на основі реалізації розподіленої комп’ютерної системи.

До основних задач, які необхідно розв’язати для досягнення мети, входять:

– розробка ефективного механізму розподіленого зберігання та моніторингу статистичних показників;

– обґрунтування апаратної платформи для функціонування розподіленої комп’ютерної системи;

– проектування зручного користувацького інтерфейсу для введення даних про хворих за населеними пунктами, а також редагування їх властивостей;

– проектування та розробка програмного забезпечення для реалізації визначеної функціональності процесу збору та опрацювання статистичних показників захворюваності (desktop рішення);

– проектування схеми бази даних на основі реляційного підходу для зберігання та маніпулювання даними;

– реалізація алгоритмів пошуку інформації за визначеними критеріями;

– автоматизоване формування звітів визначеного зразка;

– формування графічної інтерпретації статистичних даних;

– зменшення затрат часу на виконання операцій при формуванні звітів.

2.3 Характеристика об'єкту

2.3.1 Основні задачі та функції об’єкту

Основні завдання, які покликана вирішити розподілена комп’ютерна система збору, аналізу та опрацювання даних щодо поширення захворюваності на COVID-19 полягає у формуванні бази даних статистичних показників, які повинні зберігатись на географічно-віддалених серверах. Для цього потрібно розробити схему бази даних,

визначити вимоги до програмно-апаратних платформ для зберігання та доступу до інформації, побудувати архітектуру програмного забезпечення та реалізувати алгоритми опрацювання статистичних даних.

Окрім цього, кожен обласний лабораторний центр повинен мати можливість централізованого надсилання даних у глобальний репозиторій. У випадку надмірного завантаження таких центрів, потрібно забезпечити можливість запису результатів тестування з одного лабораторного центру до бази даних іншого. Тобто, крім основної функціональності, необхідно розробити ролі і права доступу до спільного використання баз даних між 24 обласними центрами.

Розподілена комп’ютерна система збору та аналізу даних щодо поширення вірусного захворювання повинна забезпечити підвищення ефективності роботи відділу епідеміологічного спостереження. Для цього необхідно проаналізувати його організаційну структуру, структуру вхідних документів, дані, які відображаються у звітах та розробити концептуальні схеми взаємодії та розподілу доступів до даних.

Основні цілі, яких потрібно досягти при використанні розподіленої комп’ютерної системи, полягають у підвищенні продуктивності праці як самих працівників, оскільки, зменшуються затрати часу на виконання операцій, так і підприємства в цілому. Крім того, при використанні такої системи збільшиться контроль над даними, їх актуальність та об’єктивність, а також в подальшому зростає достовірність і швидкість формування прогнозу захворюваності.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Розподілена комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19 повинна забезпечувати доступ до інформації, що необхідна користувачам, згідно визначених прав доступу. Доступ до ресурсів повинен бути

авторизованим та захищеним. Крім того, основною вимогою є автоматизоване формування звітів на основі вхідних даних.

3.1.1 Вимоги до структури та функціонування системи

До структури проектованої розподіленої комп’ютерної системи входить:

– апаратне та системне програмне забезпечення для функціонування комп’ютерної системи;

– база даних підтримки процесу формування та аналізу статистичних показників;

– клієнтська частина, що повинна забезпечити зв’язок та дружній інтерфейс між користувачами та базою даних;

– сервер обробки запитів;

У загальному випадку, модель бази даних повинна відображати предметну область, а клієнтська частина – відповідати за можливості обліку даних, забезпечення їх захисту, автоматичної генерації звітів.

Функціональні вимоги, що висуваються до розподіленої комп’ютерної системи:

– можливість введення, редагування та видалення даних щодо захворюваності за населеними пунктами та регіонами;

– можливість пошуку статистичних показників за вказаними критеріями;

– можливість сортування даних за визначеними критеріями;

– можливість запобігання неавторизованому доступу;

– можливість формування звітів;

– можливість керування правами доступу до інформаційних ресурсів;

– розподіл прав доступу;

– візуалізація звітних даних.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Зв’язок між компонентами системи: «база даних – клієнтська частина», здійснюється за допомогою рівня операційної системи. При умові використання

архітектури «клієнт-сервер», зв’язок між базою даних та клієнтською частиною здійснюється аналогічним чином, але з використанням локальної комп’ютерної мережі на апаратному рівні.

3.1.3 Вимоги по діагностуванню системи

Діагностика системи відбувається згідно з планом супроводу та обслуговування розподіленої комп’ютерної системи.

3.1.4 Перспективи розвитку, модернізація системи

Перспективи розвитку системи включають перехід на іншу архітектуру бази даних, але при цьому її логічна структура не повинна змінюватися. Модернізація існуючої системи можлива при розширенні функціональності системи без кардинальних змін в існуючих схемах зв’язків між розподіленими компонентами та зв’язків між таблицями у базі даних.

У випадку виявлення або зміни сутностей предметної області, система повинна адекватно реагувати на внесення змін без пошкодження існуючих даних, а також повинні бути наявні механізми та інструменти резервного копіювання та відновлення даних.

3.1.5 Вимоги до надійності системи

Розподілена комп’ютерна система повинна бути захищена на рівні операційної системи сервера та авторизованого доступу до бази даних. Надійність системи повинна забезпечуватись також і у випадку збою роботи апаратного забезпечення.

3.1.6 Вимоги до функцій та задач, які виконує система

Вимоги до функцій та задач, які виконує комп’ютерна система збору та аналізу даних щодо захворюваності на COVID-19, включають:

– забезпечення зв’язку клієнтської частини з базою даних;
– надання точних та адекватних результатів на запит користувачів;

– забезпечення часової ефективності роботи комп’ютерної системи;

– забезпечення контролю над доступом до інформації;

– забезпечення зручності використання апаратного і програмного забезпечення комп’ютерної системи;

– можливість розгортання та формування резервних копій бази даних.

3.1.7 Вимоги до апаратного забезпечення

Вимоги до сервера:

– процесор – тактова частота не менше 2,4 ГГц з кількістю логічних ядер не менше 8;

– об'єм оперативної пам'яті - не менше 16 Гб;

– об'єм жорсткого диску - не менше 1 Тб.

Вимоги до робочих станцій:

– процесор – тактова частота не менше 2,1 ГГц з кількістю логічних ядер не менше ;

– об'єм оперативної пам'яті - не менше 8 Гб;

– об'єм жорсткого диску – не менше 240 Гб.

Периферійні пристрої:

– принтер HP 1010 – 2 шт.

3.1.8 Вимоги до програмного забезпечення

Програмне забезпечення сервера – операційна система Windows Server 2019.

Програмне забезпечення робочих станцій – Windows 10 та прикладне програмне забезпечення, що використовується для забезпечення необхідного рівня продуктивності праці.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1 Архітектура розподіленої комп’ютерної системи.

2 Структура локальних вузлів розподіленої комп’ютерної системи.

3 Архітектура програмного забезпечення збору та аналізу даних щодо захворюваності на COVID-19.

4 ER-діаграма бази даних;

5 Use-case діаграма взаємодії користувача з базою даних;

6 Алгоритм роботи програмної складової комп’ютерної системи;

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка та затвердження технічного завдання	10.02-20.02.2021
2	Аналіз технічного завдання	20.02-28.02.2021
3	Визначення вимог до апаратного та програмного забезпечення комп’ютерної системи	28.02-15.03.2021
4	Проектування та реалізація схеми бази даних	15.03-30.03.2021
5	Визначення структурних модулів клієнтської частини та їх реалізація	30.03-07.05.2021
6	Розробка інструкцій з інсталяції та налаштування розподіленої комп’ютерної системи	07.05-25.05.2021
7	Безпека життєдіяльності	25.05-03.06.2021
8	Основи охорони праці	03.06-12.06.2021
9	Оформлення кваліфікаційної роботи	12.06-16.06.2021
10	Попередній захист кваліфікаційної роботи	16.06-20.06.2021
11	Захист кваліфікаційної роботи	20.06-28.06.2021

Додаток Б.

Скрипт формування бази даних

```
/*create database covidTest*/
create table CountryPart
(
ID_Part int not null primary key identity (1,1),
Part_Name varchar (150),
[Description] varchar (MAX)
)
create table District
(
ID_District int not null primary key identity (1,1),
ID_Part int foreign key references CountryPart (ID_Part),
District_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
create table Region
(
ID_Region int not null primary key identity (1,1),
ID_District int foreign key references District (ID_District),
Region_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
create table LocalityType
(
ID_Type int not null primary key identity (1,1),
LType_Name varchar (50)
)
create table Locality
(
ID_Locality int not null primary key identity (1,1),
ID_Region int foreign key references District (ID_District),
ID_Type int foreign key references LocalityType (ID_Type),
Locality_Name varchar (150),
People_Number int,
[Description] varchar (MAX)
)
create table Org_Type
(
ID_Org_type int not null primary key identity (1,1),
Org_Type_Name varchar (100),
Org_Form varchar (100)
)

create table Organization
(
ID_Organization int primary key  not null identity (1,1),
```

```
ID_Locality int foreign key references Locality(ID_Locality),
ID_Org_Type int foreign key references Org_Type(ID_Org_Type),
Org_Name varchar (MAX),
[Address] varchar (max),
[Owner] varchar (150),
[Description] varchar (MAX)
)

create table Patient
(
ID_Patient int not null primary key identity (1,1),
ID_Organization int foreign key references
Organization(ID_Organization),
FirstName varchar (100),
MiddleName varchar (150),
LastName varchar (100),
PhoneNumber varchar (20),
Passport varchar (15),
Age int
)
create table TypeTest
(
ID_TTest int not null primary key identity (1,1),
[Type_Name] varchar (200),
)
create table Test
(
ID_Test int not null primary key identity (1,1),
Test_Name varchar (200),
TestValue int,
ID_TType int foreign key references TypeTest (ID_TTest),
ID_Patient int foreign key references Patient(ID_Patient),
ID_Parent_Test int foreign key references Test(ID_Test)
)


create table ReportsType
(
ID_RType int not null primary key identity (1,1),
RType_Name varchar (200),
[Description] varchar (max)
)

create table ReportPeriod
(
ID_R_Period int not null primary key identity (1,1),
Period_Name varchar (200),
StartDate datetime,
EndDate datetime,
[Description] varchar (max)
)
create table Report
```

```
(
ID_Report int not null primary key identity (1,1),
ReportName varchar (max),
ID_R_Period int foreign key references ReportPeriod(ID_R_Period),
ID_RType int foreign key references ReportsType (ID_RType),
ReportDate datetime
)
create table StatisticGroup
(
ID_SGroup int not null primary key identity (1,1),
SGroupName varchar (max),
ID_Parent_SGroup int foreign key references StatisticGroup(ID_SGroup)
)
create table ReportDetails
(
ID_ReportDetails int not null primary key identity (1,1),
ID_Person int foreign key references Patient (ID_Patient),
ID_Report int foreign key references Report (ID_Report),
ID_SGroup int foreign key references StatisticGroup(ID_SGroup),
ID_Parent_test int foreign key references Test(ID_Test),
PeroidValue decimal,
[Description] varchar (MAX)
)
```

Додаток В.

Фрагменти програмного коду прикладного додатку

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DomainLogic.Abstracts;
using DomainLogic.Concrets;
using DomainLogic.Models;

namespace ISSClient
{
    public partial class Start : Form
    {
        public Start()
        {
            InitializeComponent();
        }

        private IList<SchemaItem> schemas
        {
            get
            {
                return
RepositoryFactory.GetSchemaItemRepository().SchemaItemsList;
            }
        }

        private void Start_Load(object sender, EventArgs e)
        {
            try
            {
                refreshDataSources();
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }
        }

        private void refreshDataSources()
        {
            schemaItemBindingSource.DataSource = schemas;
            lbCatalogs.DataSource = schemas;
        }
```

```
        private void btCreateNew_Click(object sender, EventArgs e)
        {
            CreateDbVariantForm csForm = new CreateDbVariantForm();
            csForm.ShowDialog();
            refreshDataSources();
        }

        private void btEdit_Click(object sender, EventArgs e)
        {
            if (lbCatalogs.SelectedIndex >= 0)
            {
                CreateSchema           csForm           =           new
CreateSchema((Guid)lbCatalogs.SelectedValue);
                csForm.ShowDialog();
                refreshDataSources();
            }
        }

        private void btView_Click(object sender, EventArgs e)
        {
            if (lbCatalogs.SelectedIndex >= 0)
            {
                CatalogueView            cv            =            new
CatalogueView((Guid)lbCatalogs.SelectedValue, SearchType.NoSearch);
                cv.ShowDialog();
            }
        }

        private  void  lbCatalogs_SelectedIndexChanged(object  sender,
EventArgs e)
        {
            if (lbCatalogs.SelectedIndex < 0)
                return;
            schemaItemBindingSource.CurrencyManager.Position          =
lbCatalogs.SelectedIndex;
        }

        private     IEntityItemsRepository     entitiesRepository     =
RepositoryFactory.GetEntityItemsRepository();

        private void btDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Дана дія видалить базу даних та всі
звязані     записи!     Продовжити?",     "Видалення     бази     даних",
MessageBoxButtons.YesNo,           MessageBoxIcon.Warning)           ==
System.Windows.Forms.DialogResult.Yes)
            {
                var currDatabase = lbCatalogs.SelectedItem;
                //MessageBox.Show((currDatabase is SchemaItem) ? "ss" :
"nn");
```

```
                var                    entityList                    =
entitiesRepository.GetEntitiesForSchemaId((currDatabase              as
SchemaItem).SchemaId).ToList();
                for (int i = 0; i < entityList.Count; i++)
                {

entitiesRepository.DeleteEntity(entityList[i].EntityItemId);
                }

RepositoryFactory.GetSchemaItemRepository().DeleteSchema(currDatabase
as SchemaItem);
                refreshDataSources();
            }
        }

        private void btSearch_Click(object sender, EventArgs e)
        {
            Guid schemaId = ((Guid)lbCatalogs.SelectedValue);
            SimpleSearchOptionsForm      sSearchOptForm      =      new
SimpleSearchOptionsForm(schemaId);
            sSearchOptForm.ShowDialog();
        }

        private void btClose_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Ви дійно бажаєте закрити програму?",
"Завершення роботи", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
== System.Windows.Forms.DialogResult.Yes)
            {
                this.Close();
            }
        }

        private void btExtSearch_Click(object sender, EventArgs e)
        {
            ExtendedSearchForm esf = new ExtendedSearchForm();
            esf.ShowDialog();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ISSClient
{
```

```
    public partial class CreateDbVariantForm : Form
    {
        public CreateDbVariantForm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            CreateSchema csForm = new CreateSchema(Guid.Empty);
            this.Hide();
            csForm.ShowDialog();
            this.Close();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DomainLogic.Concrets;
using DomainLogic.Models;

namespace ISSClient
{
    public partial class CreateSchema : Form
    {
        public CreateSchema(Guid schemaId)
        {
            if (schemaId == Guid.Empty)
            {
                CurrentSchemaItem  =  new  SchemaItem("Тестова  база",
"Тестовий запис");
            }
            else
            {
                CurrentSchemaItem                                    =
RepositoryFactory.GetSchemaItemRepository().GetSchemaItemBySchemaId(sc
hemaId);
            }
            InitializeComponent();
            CurrentSchemaItem.PropertyChanged          +=          new
PropertyChangedEventHandler(CurrentSchemaItem_PropertyChanged);
        }

        public  void  CurrentSchemaItem_PropertyChanged(object  sender,
PropertyChangedEventArgs e)
        {
```

```
            hasChanges = true;
        }

        public SchemaItem CurrentSchemaItem { get; private set; }

        private void CreateSchema_Load(object sender, EventArgs e)
        {
            if (CurrentSchemaItem.SchemaId != Guid.Empty)
            {
                this.Text = "Редагування бази даних";
            }
            schemaItemBindingSource.DataSource = CurrentSchemaItem;
            foreach (var entityProperty in
CurrentSchemaItem.PropertyList)
            {
                propertyList.Add(entityProperty);
            }
            entityPropertyBindingSource.DataSource = propertyList;
            dataTypeBindingSource.DataSource =
RepositoryFactory.GetDataTypesRepository().DataTypesList;
        }

        private BindingList<EntityProperty> propertyList = new
BindingList<EntityProperty>();

        private void btAddField_Click(object sender, EventArgs e)
        {
            saveProperties();
            entityPropertyBindingSource.DataSource = propertyList;
        }

        private void saveProperties()
        {
            var dataTypeRepository =
RepositoryFactory.GetDataTypesRepository();
            foreach (var property in propertyList)
            {
                CurrentSchemaItem.SaveProperty(new EntityProperty()
                {
                    Name = property.Name,
                    TypeName = property.TypeName,
                    DisplayTypeName =
dataTypeRepository.GetDisplayNameByTypeName(property.TypeName)
                });
            }
        }

        private void btSaveSchema_Click(object sender, EventArgs e)
        {
            if (propertyList.Count == 0)
            {
```

```
                MessageBox.Show("Необхідно додати хоча б одне поле для елемента бази!", "Збереження бази", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                saveProperties();
                saveSchema();
                hasChanges = false;
            }
        }

        private void saveSchema()
        {
            var repository = RepositoryFactory.GetSchemaItemRepository();
            repository.SaveSchemaItem(CurrentSchemaItem);
        }

        private void btAddNew_Click(object sender, EventArgs e)
        {
            entityPropertyBindingSource.AddNew();
        }

        private void btDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Ця дія видалить поле та всі звязані із ним дані. Продовжити?", "Видалення поля", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
                == System.Windows.Forms.DialogResult.Yes)
            {
                var cur = entityPropertyBindingSource.Current as EntityProperty;
                CurrentSchemaItem.RemoveProperty(cur.PropertyId);
                entityPropertyBindingSource.RemoveCurrent();
            }
        }

        protected override void OnClosing(CancelEventArgs e)
        {
            if (hasChanges)
            {
                var result = MessageBox.Show("Зберегти зміни перед закриттям?", "Закриття форми", MessageBoxButtons.YesNoCancel);
                switch (result)
                {
                    case DialogResult.Cancel:
                        e.Cancel = true;
                        break;
                    case DialogResult.No:
                        break;
                    case DialogResult.Yes:
```

```
                        saveProperties();
                        saveSchema();
                        break;
                    default:
                        break;
                }
            }
            base.OnClosing(e);
        }

        private bool hasChanges = false;

        private void btClose_Click(object sender, EventArgs e)
        {
            this.Close();
            //ActiveForm.Close();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DomainLogic.Concrets;
using DomainLogic.Models;
using ISSControls;

namespace ISSClient
{
    public partial class DisplayItemForm : Form
    {
        public DisplayItemForm(Guid elementId)
        {
            this.elementId = elementId;
            innerItem                                                    =
RepositoryFactory.GetEntityItemsRepository().GetEntityById(elementId);
            InitializeComponent();
            itemDisplay1.PropertyValueChanged          +=          new
EventHandler<PropertyValueChangedArgs>(itemDisplay1_PropertyValueChang
ed);
        }

        private Guid elementId;

        private EntityItem innerItem;

        private    void    lbCatalogueName_TextChanged(object    sender,
EventArgs e)
```

```
        {
            lbItemName.Location = new Point(lbCatalogueName.Location.X
+ lbCatalogueName.Width, lbCatalogueName.Location.Y);
        }

        private void DisplayItemForm_Load(object sender, EventArgs e)
        {
            var                         schema                         =
RepositoryFactory.GetSchemaItemRepository().GetSchemaItemBySchemaId(in
nerItem.SchemaId);
            itemDisplay1.InnerItem = innerItem;
            schemaItemBindingSource.DataSource = schema;
            entityItemBindingSource.DataSource = innerItem;
        }

        private void btSave_Click(object sender, EventArgs e)
        {

RepositoryFactory.GetEntityItemsRepository().SaveEntityItem(innerItem)
;
        }

        private  void  itemDisplay1_PropertyValueChanged(object  sender,
PropertyValueChangedArgs e)
        {
            innerItem[e.PropertyName] = e.NewValue;
        }

        private void btDelete_Click(object sender, EventArgs e)
        {
            if  (MessageBox.Show("Ви  справді  бажаєте  знищити  даний
запис?",       "Видалення       запису",       MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) == System.Windows.Forms.DialogResult.Yes)
            {

RepositoryFactory.GetEntityItemsRepository().DeleteEntity(innerItem.En
tityItemId);
                this.Close();
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DomainLogic.Concrets;
using DomainLogic.Models;
```

```
namespace ISSClient
{
    public partial class ExtendedSearchForm : Form
    {
        public ExtendedSearchForm()
        {
            InitializeComponent();
        }

        private   void   comboBox1_SelectedIndexChanged(object   sender,
EventArgs e)
        {
            var                        selectedId                        =
Guid.Parse(comboBox1.SelectedValue.ToString());
            var                          schema                          =
RepositoryFactory.GetSchemaItemRepository().GetSchemaItemBySchemaId(se
lectedId);
            itemDisplay1.InnerItem = EntityItem.ForSchema(schema);
        }

        private void ExtendedSearchForm_Load(object sender, EventArgs e)
        {
            schemaItemBindingSource.DataSource                           =
RepositoryFactory.GetSchemaItemRepository().SchemaItemsList;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var                        selectedId                        =
Guid.Parse(comboBox1.SelectedValue.ToString());
            var                          schema                          =
RepositoryFactory.GetSchemaItemRepository().GetSchemaItemBySchemaId(se
lectedId);
            CatalogueView   cv   =   new   CatalogueView(schema.SchemaId,
SearchType.Likeness, true, String.Empty, itemDisplay1.InnerItem);
            this.Hide();
            cv.ShowDialog();
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using DomainLogic.Concrets;
```

```
using DomainLogic.Models;
using ISSControls;

namespace ISSClient
{
    public partial class SearchResultsForm : Form
    {
        public  SearchResultsForm(IEnumerable<Guid>  schemasIds,  string
searchItem)
        {
            this.schemasIds = schemasIds;
            this.searchItem = searchItem;
            InitializeComponent();
        }

        private string searchItem;

        private IEnumerable<Guid> schemasIds;

        private SchemaItem schema;

        private List<EntityItem> entityList = new List<EntityItem>();

        private  IDictionary<Guid, IEnumerable<EntityItem>> dict  =  new
Dictionary<Guid, IEnumerable<EntityItem>>();

        //private IEnumerable<EntityItem> entityList;
        //private     List<EntityItem>     entitiesList     =     new
List<EntityItem>();
        private void addSearchResult()
        {
            int controlX = 20;
            int controlY = 0;
            int counter = 0;
            dict                                                        =
RepositoryFactory.GetEntityItemsRepository().SearchInAllDatabase(searc
hItem, schemasIds);
            foreach (var schemaId in dict.Keys)
            {
                var values = dict[schemaId];
                if (values != null && values.Count() > 0)
                {
                    DisplayResult dispRes = new DisplayResult()
                    {
                        Schema                                          =
RepositoryFactory.GetSchemaItemRepository().GetSchemaItemBySchemaId(sc
hemaId),
                        EntityList = values,
                        Location = new Point(controlX, controlY),
                        Anchor = AnchorStyles.Top | AnchorStyles.Left |
AnchorStyles.Right,
```

```
                        Size = new Size(panel1.Width - 20,
values.Count() + 1 * 30 + 150)
                    };
                    dispRes.OpenForViewClick += new
EventHandler<OpenForViewArgs>(dispRes_OpenForViewClick);
                    panel1.Controls.Add(dispRes);
                    controlY = controlY + 200;
                    counter++;
                }
            }
            if (counter == 0)
            {
                Label label = new Label() { Text = "За вашим запитом не
знайдено нічого", Width = 400, Location = new Point(50, 50) };
                label.Font = new
System.Drawing.Font(FontFamily.GenericMonospace, 12);
                panel1.Controls.Add(label);
            }
        }

        private void dispRes_OpenForViewClick(object sender,
OpenForViewArgs e)
        {
            CatalogueView cv = new CatalogueView(e.SchemaId,
SearchType.NoSearch);
            cv.Show();
        }

        private void SearchResultsForm_Load(object sender, EventArgs e)
        {
            addSearchResult();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

namespace ISSClient
{
    partial class Start
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
```

```
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(Start));
            this.lbCatalogs = new System.Windows.Forms.ListBox();
            this.schemaItemBindingSource = new
System.Windows.Forms.BindingSource(this.components);
            this.btCreateNew = new System.Windows.Forms.Button();
            this.btEdit = new System.Windows.Forms.Button();
            this.btView = new System.Windows.Forms.Button();
            this.btDelete = new System.Windows.Forms.Button();
            this.btSearch = new System.Windows.Forms.Button();
            this.btExtSearch = new System.Windows.Forms.Button();
            this.btSettings = new System.Windows.Forms.Button();
            this.gbInfo = new System.Windows.Forms.GroupBox();
            this.tbInfo = new System.Windows.Forms.TextBox();
            this.btExport = new System.Windows.Forms.Button();
            this.btClose = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.schemaItemBindingSour
ce)).BeginInit();
            this.gbInfo.SuspendLayout();
            this.SuspendLayout();
            //
            // lbCatalogs
            //
            this.lbCatalogs.DataSource = this.schemaItemBindingSource;
            this.lbCatalogs.DisplayMember = "FullDisplayName";
            this.lbCatalogs.Font = new System.Drawing.Font("Microsoft
Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.lbCatalogs.FormattingEnabled = true;
            this.lbCatalogs.ItemHeight = 20;
```

```
            this.lbCatalogs.Location  =  new  System.Drawing.Point(12,
22);
            this.lbCatalogs.Name = "lbCatalogs";
            this.lbCatalogs.Size = new System.Drawing.Size(345, 304);
            this.lbCatalogs.TabIndex = 0;
            this.lbCatalogs.ValueMember = "SchemaId";
            this.lbCatalogs.SelectedIndexChanged        +=        new
System.EventHandler(this.lbCatalogs_SelectedIndexChanged);
            //
            // schemaItemBindingSource
            //
            this.schemaItemBindingSource.DataSource                  =
typeof(DomainLogic.Models.SchemaItem);
            //
            // btCreateNew
            //
            this.btCreateNew.Image                                   =
((System.Drawing.Image)(resources.GetObject("btCreateNew.Image")));
            this.btCreateNew.ImageAlign                              =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btCreateNew.Location  =  new  System.Drawing.Point(374,
22);
            this.btCreateNew.Name = "btCreateNew";
            this.btCreateNew.Size = new System.Drawing.Size(145, 36);
            this.btCreateNew.TabIndex = 1;
            this.btCreateNew.Text = "Створити базу";
            this.btCreateNew.UseVisualStyleBackColor = true;
            this.btCreateNew.Click               +=               new
System.EventHandler(this.btCreateNew_Click);
            //
            // btEdit
            //
            this.btEdit.Image                                        =
((System.Drawing.Image)(resources.GetObject("btEdit.Image")));
            this.btEdit.ImageAlign                                   =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btEdit.Location = new System.Drawing.Point(374, 69);
            this.btEdit.Name = "btEdit";
            this.btEdit.Size = new System.Drawing.Size(145, 36);
            this.btEdit.TabIndex = 2;
            this.btEdit.Text = "Редагувати базу";
            this.btEdit.UseVisualStyleBackColor = true;
            this.btEdit.Click                  +=                  new
System.EventHandler(this.btEdit_Click);
            //
            // btView
            //
            this.btView.Image                                        =
((System.Drawing.Image)(resources.GetObject("btView.Image")));
            this.btView.ImageAlign                                   =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btView.Location = new System.Drawing.Point(374, 223);
```

```
            this.btView.Name = "btView";
            this.btView.Size = new System.Drawing.Size(145, 36);
            this.btView.TabIndex = 3;
            this.btView.Text = "Робота з даними";
            this.btView.UseVisualStyleBackColor = true;
            this.btView.Click                  +=                  new
System.EventHandler(this.btView_Click);
            //
            // btDelete
            //
            this.btDelete.Image                                     =
((System.Drawing.Image)(resources.GetObject("btDelete.Image")));
            this.btDelete.ImageAlign                                =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btDelete.Location   =   new   System.Drawing.Point(374,
116);
            this.btDelete.Name = "btDelete";
            this.btDelete.Size = new System.Drawing.Size(145, 36);
            this.btDelete.TabIndex = 4;
            this.btDelete.Text = "Видалити базу";
            this.btDelete.UseVisualStyleBackColor = true;
            this.btDelete.Click                 +=                  new
System.EventHandler(this.btDelete_Click);
            //
            // btSearch
            //
            this.btSearch.Image                                     =
((System.Drawing.Image)(resources.GetObject("btSearch.Image")));
            this.btSearch.ImageAlign                                =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btSearch.Location   =   new   System.Drawing.Point(374,
270);
            this.btSearch.Name = "btSearch";
            this.btSearch.Size = new System.Drawing.Size(145, 36);
            this.btSearch.TabIndex = 5;
            this.btSearch.Text = "Пошук в базі";
            this.btSearch.UseVisualStyleBackColor = true;
            this.btSearch.Click                 +=                  new
System.EventHandler(this.btSearch_Click);
            //
            // btExtSearch
            //
            this.btExtSearch.Image                                  =
((System.Drawing.Image)(resources.GetObject("btExtSearch.Image")));
            this.btExtSearch.ImageAlign                             =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btExtSearch.Location  =  new  System.Drawing.Point(374,
317);
            this.btExtSearch.Name = "btExtSearch";
            this.btExtSearch.Size = new System.Drawing.Size(145, 36);
            this.btExtSearch.TabIndex = 6;
            this.btExtSearch.Text = "Розширений пошук";
```

```
            this.btExtSearch.UseVisualStyleBackColor = true;
            this.btExtSearch.Click                +=                 new
System.EventHandler(this.btExtSearch_Click);
            //
            // btSettings
            //
            this.btSettings.Image                                        =
((System.Drawing.Image)(resources.GetObject("btSettings.Image")));
            this.btSettings.ImageAlign                                   =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btSettings.Location  =  new  System.Drawing.Point(374,
360);
            this.btSettings.Name = "btSettings";
            this.btSettings.Size = new System.Drawing.Size(145, 36);
            this.btSettings.TabIndex = 7;
            this.btSettings.Text = "Налаштування";
            this.btSettings.UseVisualStyleBackColor = true;
            //
            // gbInfo
            //
            this.gbInfo.Controls.Add(this.tbInfo);
            this.gbInfo.Location = new System.Drawing.Point(13, 329);
            this.gbInfo.Name = "gbInfo";
            this.gbInfo.Size = new System.Drawing.Size(344, 108);
            this.gbInfo.TabIndex = 8;
            this.gbInfo.TabStop = false;
            this.gbInfo.Text = "Інформація";
            //
            // tbInfo
            //
            this.tbInfo.BackColor                                        =
System.Drawing.SystemColors.Control;
            this.tbInfo.DataBindings.Add(new
System.Windows.Forms.Binding("Text",      this.schemaItemBindingSource,
"Information", true));
            this.tbInfo.Enabled = false;
            this.tbInfo.Location = new System.Drawing.Point(6, 19);
            this.tbInfo.Multiline = true;
            this.tbInfo.Name = "tbInfo";
            this.tbInfo.Size = new System.Drawing.Size(331, 83);
            this.tbInfo.TabIndex = 0;
            //
            // btExport
            //
            this.btExport.Image                                          =
global::ISSClient.Properties.Resources.boxDownload;
            this.btExport.ImageAlign                                     =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btExport.Location   =   new   System.Drawing.Point(374,
159);
            this.btExport.Name = "btExport";
            this.btExport.Size = new System.Drawing.Size(145, 36);
```

```
            this.btExport.TabIndex = 9;
            this.btExport.Text = "Експорт бази";
            this.btExport.UseVisualStyleBackColor = true;
            //
            // btClose
            //
            this.btClose.Image                                          =
global::ISSClient.Properties.Resources.block_16;
            this.btClose.ImageAlign                                     =
System.Drawing.ContentAlignment.MiddleLeft;
            this.btClose.Location = new System.Drawing.Point(374, 403);
            this.btClose.Name = "btClose";
            this.btClose.Size = new System.Drawing.Size(145, 36);
            this.btClose.TabIndex = 10;
            this.btClose.Text = "Завершити роботу";
            this.btClose.UseVisualStyleBackColor = true;
            this.btClose.Click                  +=                  new
System.EventHandler(this.btClose_Click);
            //
            // Start
            //
            this.AutoScaleDimensions  =  new  System.Drawing.SizeF(6F,
13F);
            this.AutoScaleMode                                          =
System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(533, 449);
            this.Controls.Add(this.btClose);
            this.Controls.Add(this.btExport);
            this.Controls.Add(this.gbInfo);
            this.Controls.Add(this.btSettings);
            this.Controls.Add(this.btExtSearch);
            this.Controls.Add(this.btSearch);
            this.Controls.Add(this.btDelete);
            this.Controls.Add(this.btView);
            this.Controls.Add(this.btEdit);
            this.Controls.Add(this.btCreateNew);
            this.Controls.Add(this.lbCatalogs);
            this.FormBorderStyle                                        =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon                                                   =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximizeBox = false;
            this.Name = "Start";
            this.StartPosition                                          =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Інфомаційно - пошукова система 1.0 - Старт";
            this.Load += new System.EventHandler(this.Start_Load);

((System.ComponentModel.ISupportInitialize)(this.schemaItemBindingSour
ce)).EndInit();
            this.gbInfo.ResumeLayout(false);
            this.gbInfo.PerformLayout();
```

```
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.ListBox lbCatalogs;
        private                     System.Windows.Forms.BindingSource
schemaItemBindingSource;
        private System.Windows.Forms.Button btCreateNew;
        private System.Windows.Forms.Button btEdit;
        private System.Windows.Forms.Button btView;
        private System.Windows.Forms.Button btDelete;
        private System.Windows.Forms.Button btSearch;
        private System.Windows.Forms.Button btExtSearch;
        private System.Windows.Forms.Button btSettings;
        private System.Windows.Forms.GroupBox gbInfo;
        private System.Windows.Forms.TextBox tbInfo;
        private System.Windows.Forms.Button btExport;
        private System.Windows.Forms.Button btClose;
    }
}
```