МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ,
ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ ТА РАДІОЕЛЕКТРОННИХ СИСТЕМ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Віктор ГНАТЮК
“_____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

Тема: «Метод інтеграції реляційних баз даних із використанням платформи D2RQ».

Виконавець: _____ Андрій СТЕПАНЮК
(підпис)

Керівник: _____ Віталій КУРУШКІН
(підпис)

Консультанти з окремих розділів пояснювальної записки:

Консультант розділу «Охорона праці» _____ Батир ХАЛМУРАДОВ
(підпис)

Консультант розділу «Охорона навколишнього середовища»
_____ Андріан ЯВНЮК
(підпис)

Нормоконтролер: _____ Денис БАХТІЯРОВ
(підпис)

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра телекомунікаційних та радіоелектронних систем
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Віктор ГНАТЮК
“_____”_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Степанюка Андрія Віталійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи: «Метод інтеграції реляційних баз даних із використанням платформи D2RQ»
затверджена наказом ректора від «28» вересня 2023 р. №1965/ст
2. Термін виконання роботи: з 02.10.2023 р. по 31.12.2023 р.
3. Вихідні дані до роботи: база даних
4. Зміст пояснювальної записки: аналітичний огляд предметної області, аналіз підходів для організації баз даних, метод інтеграції реляційних баз даних із використанням платформи D2RQ, програмна реалізація методу інтеграції реляційних баз даних
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: слайди презентації до доповіді на захисті кваліфікаційної роботи

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів кваліфікаційної роботи	02.10.2023-04.10.2023	Виконано
2	Вступ	05.10.2023-08.10.2023	Виконано
3	Аналітичний огляд предметної області	09.10.2023-22.10.2023	Виконано
4	Аналіз підходів для організації баз даних	23.10.2023-05.11.2023	Виконано
5	Метод інтеграції реляційних баз даних із використанням платформи D2RQ	06.11.2023-10.11.2023	Виконано
	Програмна реалізація методу інтеграції реляційних баз даних	11.11.2023-30.11.2023	Виконано
6	Охорона праці	01.12.2023-06.12.2023	Виконано
7	Охорона навколишнього середовища	07.12.2023-17.12.2023	Виконано
8	Усунення недоліків та захист кваліфікаційної роботи	18.12.2023-31.12.2023	Виконано

7. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона праці	к.м.н., професор Батир ХАЛМУРАДОВ		
Охорона навколишнього середовища	к.б.н., доц. Андріан ЯВНЮК		

8. Дата видачі завдання: “29" вересня 2023 р.

Керівник кваліфікаційної роботи _____ Віталій КУРУШКІН
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Андрій СТЕПАНЮК
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Кваліфікаційна робота «Метод інтеграції реляційних баз даних із використанням платформи D2RQ» містить 87 сторінок, 22 рисунки, 5 таблиць, 56 використаних джерел.

РЕЛЯЦІЙНА БАЗА ДАНИХ, D2RQ, МАПІНГ, RESOURCE DESCRIPTION FRAMEWORK (RDF), СЕМАНТИЧНИЙ ВЕБ, SPARQL, ПРЯМЕ ВІДОБРАЖЕННЯ, БАЗА ДАНИХ ДО RDF, RDF SCHEMA (RDFS), ONTOLOGY (ОНТОЛОГІЯ), LINKED DATA (ПОВ'ЯЗАНІ ДАНІ), JDBC, ІНТЕГРАЦІЯ, SQL, RDF QUERY LANGUAGE, WEB ONTOLOGY LANGUAGE (OWL), URI (UNIFORM RESOURCE IDENTIFIER), ТРАНСФОРМАЦІЯ ДАНИХ, RDB2RDF, МЕТОДИ ІНТЕГРАЦІЇ.

Мета кваліфікаційної роботи: розробити та дослідити методи та моделі інтеграції реляційних баз даних з використанням технологій та інструментів Semantic Web.

Об'єктом дослідження є різнорідна інформація, що зберігається в реляційних баз даних, а також методи та технології Semantic Web.

Предмет дослідження полягає у вивченні технічних, алгоритмічних та методологічних основ інтеграції реляційних баз даних в контексті семантичного вебу за допомогою платформи D2RQ.

Практична новизна полягає в можливості інтеграції баз даних у формат зрозумілий для машинної. Це дослідження може стати міцним підґрунтям глобального використання пошукових систем, що базуються лише на формалізованих даних. Дане дослідження, також може дозволити заощадити великим організаціям, які мають величезні бази даних.

Результати цієї роботи було докладено на конференції Молодь і сучасні інформаційні технології.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

RDB - Relational Database (Реляційна база даних).

D2RQ - Direct Mapping of Relational Data to RDF (Пряме відображення реляційних даних в RDF).

RDF - Resource Description Framework (Фреймворк опису ресурсів).

RDFS - RDF Schema (Схема RDF).

SPARQL - SPARQL Protocol and RDF Query Language (Протокол SPARQL та мова запитів RDF).

DBMS - Database Management System (Система управління базами даних).

SQL - Structured Query Language (Структурована мова запитів).

URI - Uniform Resource Identifier (Уніфікований ідентифікатор ресурсів).

OWL - Web Ontology Language (Мова веб-онтологій).

API - Application Programming Interface (Інтерфейс програмування додатків).

DB - Database (База даних).

R2RML - RDB to RDF Mapping Language (Мова мапінгу реляційних баз даних в RDF).

RDMS - Relational Database Management System (Система управління реляційними базами даних).

D2R - Database to RDF (Від бази даних до RDF).

RDB2RDF - Relational Database to RDF (Від реляційної бази даних до RDF).

JDBC - Java Database Connectivity (Підключення до бази даних за допомогою Java).

HTTP - HyperText Transfer Protocol (Протокол передачі гіпертексту).

IRI - Internationalized Resource Identifier (Інтернаціоналізований ідентифікатор ресурсів).

LOD - Linked Open Data (Пов'язані відкриті дані).

RDFa - RDF in attributes (RDF у атрибутах).

XML - eXtensible Markup Language (Розширювана мова розмітки).

XSD - XML Schema Definition (Опис схеми XML).

TTL - Turtle (формат представлення RDF).

REST - Representational State Transfer (Архітектурний стиль передачі стану представлення).

CRUD - Create, Read, Update, Delete (Основні операції управління даними).

ERD - Entity Relationship Diagram (Діаграма зв'язків сутностей).

ETL - Extract, Transform, Load (Процес витягу, трансформації та завантаження даних).

URIQA - URI Query Agent Model (Модель агента запиту URI).

SHACL - SHAPES Constraint Language (Мова обмеження форм).

SKOS - Simple Knowledge Organization System (Проста система організації знань).

ВСТУП

Актуальність теми. На багатьох великих підприємствах інформація зберігається в різних базах даних. Інформація цих баз даних може повторюватися або доповнювати одна одну. Робота з розподіленими джерелами баз даних складний і не зручний процес. Найчастіше, різні бази даних мають різні схеми. Це не дає змоги інтегрувати інформацію в одну велику базу даних, що містить необхідні для роботи таблиці.

Існує безліч рішень щодо способів зберігання інформації, що міститься в базах даних. Але ці рішення не позбавляють від проблеми інтеграції даних з декількох джерел. Інтеграція інформації реляційних баз даних є актуальним завданням для фахівців, пов'язаних з опрацюванням даних з різнорідних джерел.

В сучасному світі інформаційних технологій реляційні бази даних відіграють ключову роль в управлінні, зберіганні та обробці великих обсягів інформації. Однак з появою технологій семантичного вебу, що дозволяють представляти дані в графовій формі, виникла потреба в інтеграції реляційних баз даних з цими технологіями. Платформа D2RQ, яка представляє собою високоефективний механізм для мапінгу реляційних баз даних до RDF (Resource Description Framework), набула великої популярності в даному напрямку.

Водночас для вирішення завдання роботи з різнорідними даними WWW організація Всесвітньої Павутини розвиває ідею створення наступної версії WWW - Семантичної Павутини (Semantic Web). Семантична Павутина передбачає зберігання всієї інформації Всесвітньої Павутини у форматі зрозумілому і обчислювальній машині, і людині. Крім цього, цей формат надає можливість виконувати на цих даних логічні висновки. Семантична павутина існує зараз як надбудова над Всесвітньою Павутиною, але, на жаль, глобально не використовується. Однією з основних проблем впровадження Семантичної Павутини є переведення великих обсягів уже наявної інформації з реляційних баз даних у формат даних Семантичної павутини.

Для реалізації SW вже розроблено велику кількість стандартів (наприклад, RDF/RDFS, OWL, SPARQL) і різних програмних інструментів (наприклад, D2R і Protégé).

Ґрунтуючись на ідеях і стандартах SW можна дослідити можливість інтеграції реляційних БД організації.

У рамках даної роботи будуть розглянуті основні поняття та принципи роботи платформи D2RQ, методи та алгоритми мапінгу реляційних баз даних до RDF, а також практичний приклад використання D2RQ для інтеграції конкретної реляційної бази даних.

Вивчення цієї тематики актуальне, оскільки інтеграція реляційних баз даних з семантичним вебом може значно розширити можливості аналізу та використання даних, а також сприяти створенню нових, ефективніших систем управління інформацією [1-40].

Мета кваліфікаційної роботи: розробити та дослідити методи та моделі інтеграції реляційних баз даних з використанням технологій та інструментів Semantic Web.

Об'єктом дослідження є різнорідна інформація, що зберігається в реляційних баз даних, а також методи та технології Semantic Web.

Предмет дослідження полягає у вивченні технічних, алгоритмічних та методологічних основ інтеграції реляційних баз даних в контексті семантичного вебу за допомогою платформи D2RQ.

Практична новизна полягає в можливості інтеграції баз даних у формат зрозумілий для машинної. Це дослідження може стати міцним підґрунтям глобального використання пошукових систем, що базуються лише на формалізованих даних. Дане дослідження, також може дозволити заощадити великим організаціям, які мають величезні бази даних.

Результати цієї роботи було докладено на конференції Молодь і сучасні інформаційні технології.

Для комплексного аналізу теми "Метод інтеграції реляційних баз даних із використанням платформи D2RQ" використовувались наступні ***методи дослідження:***

Теоретичний аналіз:

- Вивчення наукової літератури за даною темою.
- Аналіз специфікацій та документації платформи D2RQ.
- Порівняльний аналіз D2RQ з аналогічними платформами та технологіями.

Експериментальне дослідження:

- Практичне використання D2RQ для інтеграції реальних реляційних баз даних.
- Тестування швидкості, надійності та ефективності мапінгу за допомогою D2RQ.
- Виявлення та аналіз можливих проблем та обмежень при використанні платформи.

Комп'ютерне моделювання:

- Створення моделей реляційних баз даних та їх інтеграція з семантичним вебом за допомогою D2RQ.
- Аналіз результатів моделювання з метою оптимізації процесу інтеграції.

Метод експертних оцінок:

- Залучення фахівців у сфері баз даних та семантичного вебу для оцінювання якості, ефективності та перспектив використання D2RQ.

Метод кейс-стаді (вивчення випадків):

- Аналіз реальних випадків використання D2RQ у різних організаціях та проєктах, визначення переваг та недоліків конкретних сценаріїв застосування.

Статистичний аналіз:

- Збір та обробка даних про ефективність, швидкість та надійність інтеграції за допомогою D2RQ у різних умовах.

Застосування цих методів дозволить отримати глибоке розуміння особливостей роботи платформи D2RQ, її переваг та недоліків, а також визначити оптимальні сценарії її використання для інтеграції реляційних баз даних.

Апробація отриманих результатів. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- Науково-практична конференція «Проблеми експлуатації та захисту інформаційно-комунікаційних систем», м. Київ, 2023 р.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналітичний огляд літератури

Розроблення систем керування базами даних у 90-х роках супроводжувалося розвитком технологій створення широкомасштабних (wide-area) комп'ютерних мереж. Стало само собою очевидним, що зберігання та пошук, а отже й інтеграція розподіленої інформації стає дедалі важливішим і важливішим для підприємств і розподілених (децентралізованих) організацій. З безперервним зростанням поширення технологій глобальних мереж (WAN) та Інтернету, важливість такої інтеграції почала дедалі більше зростати. Стало очевидним, що традиційні підходи, розроблені для систем мультибаз даних і розподілених баз даних, стали вже не придатними (адекватними) особливо в тих випадках, коли інтегрована інформація підтримувалася в дуже автономних і розподілених системах. Хоча багато базових алгоритмів є схожими, для того, щоб вони могли розв'язувати нові проблеми, їх потрібно розширити й об'єднати з новими поняттями. Для інтеграції інформації зазвичай застосовують підхід посередників-адаптерів, якщо інформація надається різними джерелами, такими, як різні види систем баз даних (СУБД), наборами XML-документів та/або web-сайтами і різними типами інтерфейсів, такими, як SQL, XQuery або Web-сервісами (services) [1].

1.2. Посередники в майбутніх інформаційних системах

Один із прихильників підходу на основі посередників (mediator approach), Gio Wiederhold, наступним чином сформулював поняття посередника [2]: "Посередник (mediator) це програмний модуль, який використовує закодовані знання про деякі множини або підмножини даних для того, щоб створювати інформацію для додатків вищого рівня".

Цю цитату можна використовувати як високорівневе визначення архітектури, заснованої на посередниках. Відповідно до [2], спільнота дослідників зіткнулася з двома типами проблем:

1. "для окремих баз даних основною перешкодою для доступу кінцевих користувачів є обсяг даних, що стають доступними, нестача абстракції та потреба зрозуміти представлення даних";
2. "великою проблемою під час об'єднання інформації з безлічі баз даних є невідповідність, що трапляються в поданні та структурі інформації";

Проблеми першого виду були добре вирішені протягом останньої пари десятиліть: сьогодні, не є серйозною проблемою управління і пошук інформації у великих обсягах даних. OLAP-технології є досить потужними, щоб було можливим абстрагуватися від детально деталізованих даних і дуже швидко надати більш значущу (more meaningful) інформацію. Однак досі ще залишилося проблемою правильна інтерпретація (розуміння) даних, представлених у базах даних [3].

Другий тип проблем, коротко кажучи, є основною метою поточних досліджень у галузі Semantic Web. При об'єднанні (тобто інтеграції) інформації з різних джерел звичайною проблемою є правильне узгодження їхніх семантик, особливо, якщо базові дані повинні об'єднуватися і оброблятися автоматично. Wiederhold перелічив деякі з проблем, які зустрічаються під час об'єднання автономних джерел, до яких належать:

- конфлікти імен;
- різна детальність рівня абстрагування;
- різний часовий контекст;
- відмінність у семантиці предметних областей;
- відмінність у семантиці значень, тощо.

Ба більше, він описав абстрактну модель обробки інформації, посилаючись на статтю [4] "Як ми можемо думати" ("As We May Think"). Wiederhold назвав процес отримання, опрацювання та комунікації - інформаційним плануванням, оскільки дані беруть із минулого з метою впливу на майбутнє. Він стверджує, що інтерфейси майбутніх інформаційних систем повинні брати на себе активну роль у підтримці користувачів під час такого планування. Він називає посередник "модулем, який займає

явний, активний шар між додатками користувачів і джерелами даних" і описує їхню архітектуру так, як це показано на Рисунку 1.1.

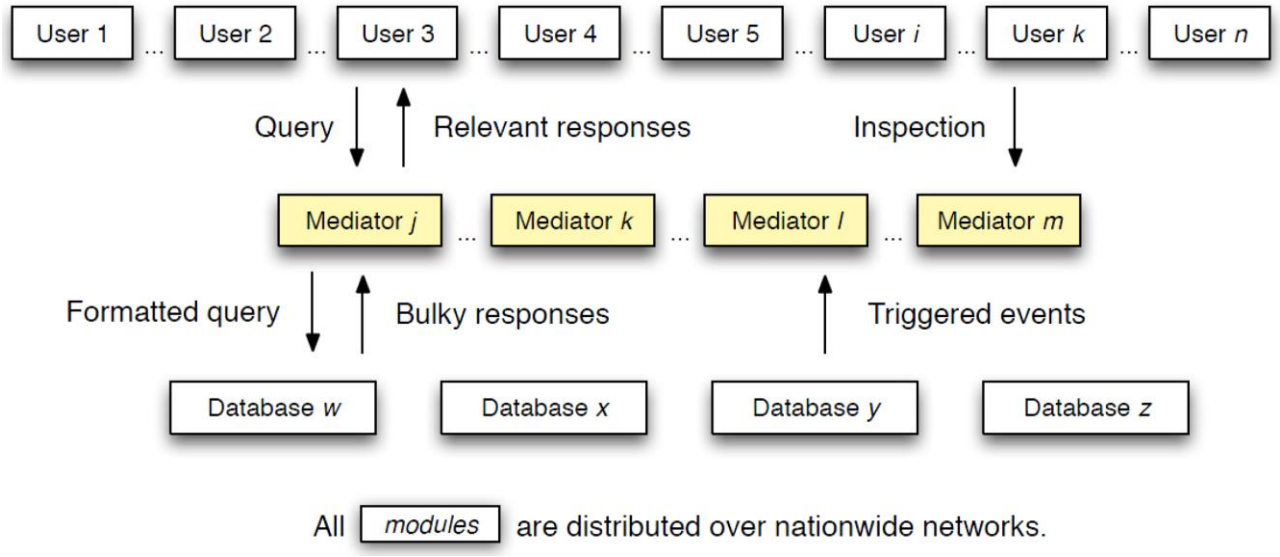

Рис. 1.1. Посередники як інтерфейси для потоків інформації [4]

Таке розуміння посередників схоже на теорію програмних агентів (software agents), які діють від імені своїх користувачів і виконують різні завдання. Наприклад, цілями систем Carnot [4] та InfoSleuth [Bayardo et al. 1997] було розроблення заснованих на агентах узагальнених середовищ зі спеціалізованими програмними агентами, відповідальними за виявлення ресурсів (resource discovery), організацію взаємодії та інтеграцію інформації і сервісів в динамічних середовищах. Підхід, заснований на інтелектуальних програмних агентах, що виконують складні завдання, які зазвичай вимагають деякого подібності (виду) інтелекту людини, є фактично вельми амбітним і важким для реалізації [5].

У контексті інтеграції інформації, посередники мають бути здатними обробляти запити шляхом аналізу довільної кількості різних відомих джерел інформації. Зазвичай це досягається шляхом розміщення на кожному джерелі даних адаптера, який є відповідальним за обробку локального запиту і перетворення даних. У наступних розділах розглядаються конкретні реалізації таких, заснованих на посередниках, систем інтеграції інформації. Здебільшого всі підходи можна розділити на проєкти, які виконують у співтоваристві дослідників баз даних (як, наприклад, система Garlic компанії

IBM), і проєкти, які виконують у співтовариствах дослідників штучного інтелекту та обчислень на основі агентів (як, наприклад, InfoSleuth).

Першим буде описано проєкт Garlic компанії IBM, оскільки він уже включав багато важливих особливостей (аспектів) типових систем, заснованих на посередниках-адаптерах.

1.3. IBM "Garlic"

Технологія IBM "Garlic", що використовує федеровані сервери для пошуку необхідної інформації про об'єкт із різних, автономних джерел [6].

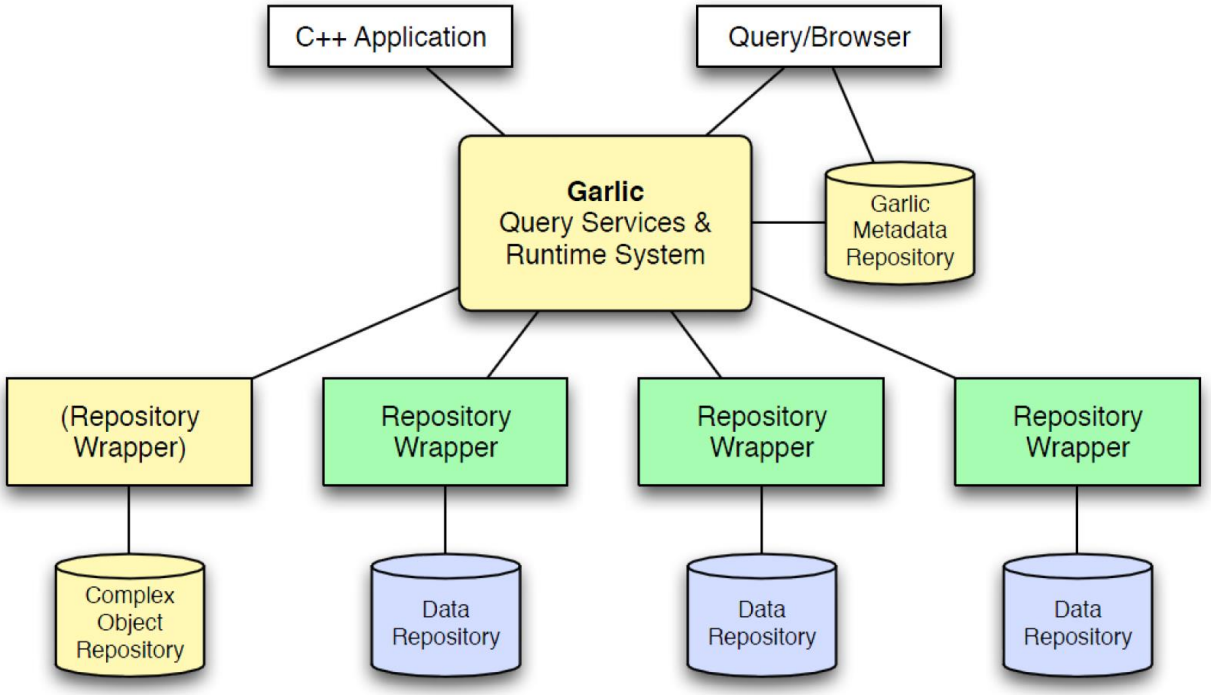

Рис. 1.2. Архітектура системи Garlic

У центральній частині рисунку 1.2. показано посередника системи Garlic, що включає сервіси запитів і систему виконання, а також сховище метаданих. Система Garlic може бути пов'язана з додатками мовою C++ за допомогою спеціального API. Інтерактивний компонент обробки запитів/перегляду надає кінцевим користувачам більш зручний спосіб виконання доступу до інтегрованої інформації. Цей компонент

був новим підходом виконання запитів, що мав назву "запити-по-графічному-шаблону". Фактично, цей підхід був дуже схожий на гіпермедіа браузери, що з'являються. Порівняно з World Wide Web, який застосовував набагато менш обмежений і не скоординований спосіб організації зв'язків і посилань, інформація в системі Garlic надходила з баз даних і була складена на основі алгебраїчних правил [6].

У нижній частині рис. 1.2 показано кілька сховищ даних, до кожного з них прикріплено адаптер, пов'язаний із системою Garlic. Робота адаптера полягає в перекладі інформації про типи і схеми даних і перетворенні запитів на рідну для джерела даних мову запитів або у виконанні API викликів до віддаленого сховища. Вся інформація про глобальну схему системи Garlic та інша інформація, пов'язана з виконанням перекладу, підтримується в сховищі метаданих. Одним із таких сховищ є спеціальне сховище: Complex Object Repository (Сховище Складних Об'єктів), яке використовується для загального зв'язування всієї інформації. Воно може розташовуватися у віддаленому сховищі (repository) або в самій системі Garlic і використовуватися для надання додаткової інформації для зв'язування споріднених об'єктів, що раніше ніяк не з'єднувалися між собою, з різних сховищ.

Архітектура системи Garlic ґрунтується на таких припущеннях [6]:

- для будь-якого виду джерела даних (реляційних баз даних різних виробників, старіших, не реляційних баз даних, як-от IMS компанії IBM, файлових систем, заснованих на записах, мультимедіа баз даних зі специфічними для мультимедіа можливостями пошуку тощо) має бути можливість написати адаптер;
- адаптер повинен використовувати можливості базової інформаційної системи
- вибір на основі ключових слів,
- інтервальні запити ,
- булеві вирази,
- зіставлення на основі подібності
  - має бути абсолютно проста можливість створити новий адаптер або розширити новими можливостями наявні адаптери (наприклад, у міру розвитку можливостей віддаленого виконання роботи (remote capabilities));

- додавання нового адаптера до цієї архітектури не повинно впливати (не повинно залежати) на інші компоненти системи.

1.4. Проєкт TSIMMIS

Проєкт TSIMMIS (абревіатура для The Stanford-IBM Manager of Multiple Information Sources) був спільним проєктом дослідницького центру Almaden Research Center компанії IBM і Stanford University у той самий час, коли в цьому центрі виконувався проєкт Garlic [7]. Однак, архітектура системи TSIMMIS багато в чому відрізняється від архітектури системи Garlic.

По-перше, в архітектурі системи TSIMMIS використовуються впорядковані в стеки посередники, кожен з яких приймає вхід від безлічі адаптерів або інших посередників і виконує певне опрацювання, щоб внести деяку додаткову вартість. У системі TSIMMIS, адаптери називаються трансляторами. Загальна схема архітектури TSIMMIS показана на рис. 1.3.

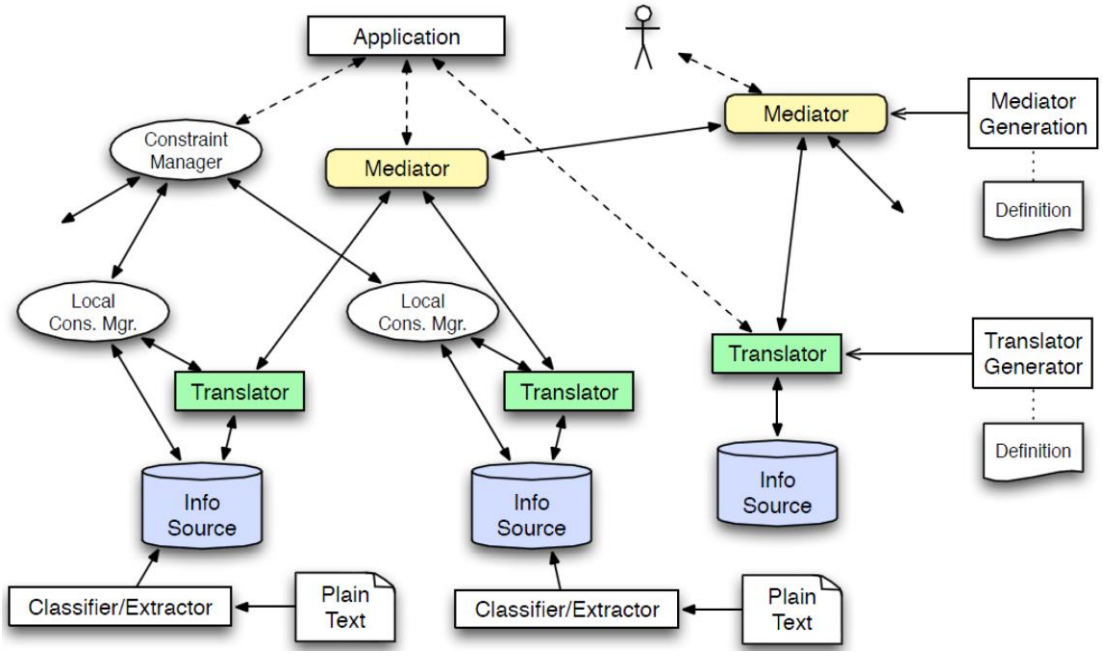

Рис. 1.3. Архітектура системи TSIMMIS

Як показано на рис. 1.3, користувачі можуть обирати один з існуючих посередників (а також безпосередньо адаптери) для виконання запитів. Посередники мають

фіксовану поведінку, яка жорстко кодується, як частина їхньої реалізації. Наприклад, посередники можуть об'єднувати два джерела даних, що мають адаптери, за допомогою таких операцій, як join, union або інших операцій [7]. У той час, як один адаптер може надавати інформацію про бібліотеку, посередник може додавати обкладинки книжок, отримані від іншого адаптера, у проміжні результати. Інший посередник може включати розширену інформацію про автора або відгуки користувачів на третьому кроці. Кожен інтерфейс запитів є чимось подібним до більш-менш складного специфічного представлення доступних елементів даних. Як буде показано далі, мова запитів у системі TSIMMIS явно не надає операторів, які є в реляційній алгебрі та інших декларативних мовах запитів. Користувач не має безпосередньої можливості описати операції join або union, це можливо тільки в тому разі, якщо немає посередника, що надає таке конкретне представлення двох інших джерел, які можуть бути посередниками або адаптерами [8]. Підхід TSIMMIS дуже чітко відповідає баченню Gio Wiederhold на архітектуру, що ґрунтується на посередниках-адаптерах, який було описано в розділі 1.1, хоча можна сперечатися, що ці можливості є доволі обмеженими, тому що зв'язування та інші типові операції мають явно забезпечуватися посередниками. Потреба у великому наборі різних посередників також стала приводом для розробників системи TSIMMIS розробляти генератори для автоматичного створення, як адаптерів, так і посередників. Адаптери та посередники можуть генеруватися на основі правил і програмних бібліотек.

По-друге, адаптери системи TSIMMIS були розширені сортувальниками та екстракторами, які здатні витягувати структуровані дані з неструктурованих джерел, таких, як простий текст або HTML документи. Система Garlic також підтримує інтеграцію мультимедіа джерел даних, але порівняно з TSIMMIS, у ній передбачається, що мультимедіа система надає спеціальний адаптер, який надає STAR і POP елементи, за допомогою яких можна безпосередньо використовувати специфічні функції доступу. Система TSIMMIS надає сортувальники та витягувачі незалежно від адаптерів. Однак, така особливість може також розглядатися як розширення, які є абсолютно незалежними від системи посередників-адаптерів. Джерела мультимедіа даних також можуть надавати власні метадані і в цьому разі не потрібні спеціальні екстрактори. На

відміну від підходу системи Garlic до інтеграції мультимедіа джерел, сортувальники та екстрактори TSIMMIS можуть прикріплятися до адаптерів будь-якої іншої системи інтеграції і тому не є складовою частиною цієї архітектури.

У системі TSIMMIS є засоби підтримки глобальних логічних обмежень за допомогою менеджерів глобальних і локальних обмежень. Однак, при інтеграції автономних джерел інформації зазвичай є мало можливості виправляти несумісності, якщо обмеження порушуються. Припускаючи, що локальні обмеження накладаються і перевіряються джерелами даних, будь-яке глобальне обмеження повинно визначатися таким чином, щоб воно жодним чином не могло порушитися.

Глобальна модель даних у системі Garlic також відрізняється від глобальної моделі даних TSIMMIS. У той час, як описи джерел даних у системі Garlic фіксовані, архітектура системи TSIMMIS є набагато гнучкішою.

Модель обміну метаданими. У роботі [9] запропоновано самоописувану глобальну модель даних, яка називається Object Exchange Model (OEM, Модель обміну об'єктами).

Цю модель можна використовувати для опису даних без необхідності на рівні опису явної схеми. Це полегшує управління доступними посередниками й адаптерами, оскільки не потрібно, щоб посередник підтримував заздалегідь описану глобальну схему. Модель OEM ґрунтується на структурованому графі, що формується вкладеними кортежами такої форми < label, type, value > [23], у якій value може бути значенням кінцевої вершини або посиланням на вкладений кортеж. Приклад цього графа наведено в таблиці 1.1

Таблиця 1.1

OEM приклад запиту [10].

Вихідна модель (посередник)	Запит:
Biblio):	SELECT bib.doc.topic
< bib, set, {doc_1, doc_2, . . . , doc_n} >	FROM Biblio
doc_1 : < doc, set, {au_1, top_1, cn1} >	WHERE
au_1 : < authors, set, {au^1_1} >	bib.doc.authors.author-ln = "Ullman"
au_{11} : < author-ln, str, “Ullman” >	
top_1 : < topic, str, “Databases” >	**Результат запита:**
cn_1 : < local-call#, integer, 25 >	< answer, set, {o_1, o_2} >
doc_2 : < doc; set, {au_2, top_2, cn_2} >	o_1 : < topic, str, “Databases” >
au_2 : < authors, set, {au^1_2, au^2_2, au^3_2} >	o_2 : < topic, str, “Algorithms” >
au^1_2 : < author-ln, str, “Aho” >	
au^2_2 : < author-ln, str, “Hopcroft” >	
au^3_2 : < author-ln; str; “Ullman” >	
top_2 : < topic, str, “Algorithms” >	
cn_2 : < dewey-decimal#, str, “BR273” >	

Семантика запиту ґрунтується на мітці. Моделювальна семантика в моделі OEM є не досконалішою, ніж у реляційній моделі: самоописувані імена використовуються для властивостей in-line під час опису даних. Немає можливості моделювати явні взаємозв'язки між класами, ієрархії класів, а також немає можливості додавати додаткові логічні обмеження, як це можна робити в мовах RDF-Schema and OWL. Мовою глобальних запитів у TSIMMIS є мова OEM-QL. З погляду структури, як OEM, так і OEM-QL [11] є навіть подібними до XML і XPath/XQuery (без використання явної XML Schema). Приклад запиту мовою OEM-QL і відповідна вихідна модель наведено на рисунку. 1.4. Як показує цей приклад, обробка запитів в TSIMMIS ґрунтується на зіставленні шляхів у графі. Для запиту, показаного на рисунку 1.4, посередник з ім'ям Biblio має знайти всі шляхи до структури під-об'єктів, описаної послідовністю {bib, doc, authors, authors-ln }, у якій останній об'єкт має значення "Ullman". Для кожного з цих шляхів, підструктура bib.doc розширюється для отримання всіх значень для bib.doc.topic.

Обробка запитів у системі TSIMMIS. Аналогічно системі Garlic, можуть бути використані вихідні можливості обробки запитів, наявні в інформаційних систем, що

мають адаптери. Однак у системі TSIMMIS використовується інша методологія, яка ґрунтується на включенні запитів. Можливості адаптера описуються параметризованими правилами [12]. У цьому сенсі, можливості (ці ж правила) також описують, які кортежі надають джерела даних. Під час опрацювання запитів ці правила зіставляються з отриманим (оброблюваним) запитом. На відміну від системи Garlic, ці правила не є описаними так докладно, як опис алгебраїчних операцій. У гіршому випадку адаптер (або посередник) можуть підтримувати тільки конкретні запити, і якщо деяка частина запиту відсутня, то адаптер не може відповісти на нього. Однак, якщо визначено набір правил, а відповідь на запит не можна отримати безпосередньо, то адаптер намагатиметься об'єднати правила для пошуку еквівалентного рішення. Цей процес аналогічний обчисленню включення запитів, що використовується для дозволу (формування) уявлень у реляційних системах управління базами даних. Оптимізатор для системи виконання TSIMMIS посередників було запропоновано в [12-15]. Це алгебраїчний оптимізатор, який є частиною розширювача подань і оптимізатор на основі вартостей, що визначає, які запити надіслано конкретним джерелам і в якому порядку їх надіслано. Він використовує підхід, що значною мірою відрізняється від підходів, які використовують звичайні, засновані на вартісності, оптимізатори для планів реляційної алгебри (наприклад, оптимізатор Starburst).

Багато в чому це пов'язано з тим, що цей підхід обробки запитів ґрунтується на правилах і значною мірою пов'язаний з логічним програмуванням. Ця оптимізація є більш важкою, оскільки модель OEM використовує вкладені об'єкти, які мають невідому структуру. Наприклад, коли посередник виконує зв'язування, то неможливо передбачити, чи буде конкретне значення отримано від того чи іншого джерела, що робить проштовхування вибірок вниз неможливим. Гнучка архітектура системи TSIMMIS, заснованої на моделі OEM і мові запитів OEMQL, вочевидь має вузьке місце, пов'язане з реалізацією і застосуванням складної оптимізації запитів. Це є основною причиною, чому цей підхід є придатним тільки до певної міри [14, 15].

Заснований на Web перегляд графів відповідей моделі OEM. Досить цікаво, як у 90-ті роки були запропоновані перші підходи до використання WWW, як основи для інтерфейсу користувачів розподілених систем. Як частину проєкту TSIMMIS

було розроблено Information Explorer (Оглядач Інформації), заснований на системі Mosaic (Mosaic-based Information Explorer, MOBIE). Хоча він був дуже

простим, порівняно з сучасними інтерфейсами, але принаймні він був першим підходом до навігації по графу взаємопов'язаної інформації на основі концепції гіпермедіа. Для пояснення сенсу OEM міток, MOBIE мав спеціальну кнопку для виклику допомоги, яку можна було використовувати для виклику текстових описів. Це дуже схоже на властивість rdfs:comment, що використовується для опису елементів RDF схеми природною мовою. Порівняно з розширюваною мовою RDF (Resource Description Framework), виразність моделі OEM є низькою. Запити мовою OEM-QL починаються з кореневої вершини, що не вимагається при виконанні запитів до RDF графів. Мова RDF краще підходить для опису розподіленої інформації, оскільки вона не використовує припущення про унікальність імен і ґрунтується на припущенні про відкритість світу.

1.5. Компонент розподіленого пошуку інформації (DISCO)

Система DISCO (Distributed Information Search Component), розроблена в Європейському проєкті [16], розроблялася, як компактна і проста система.

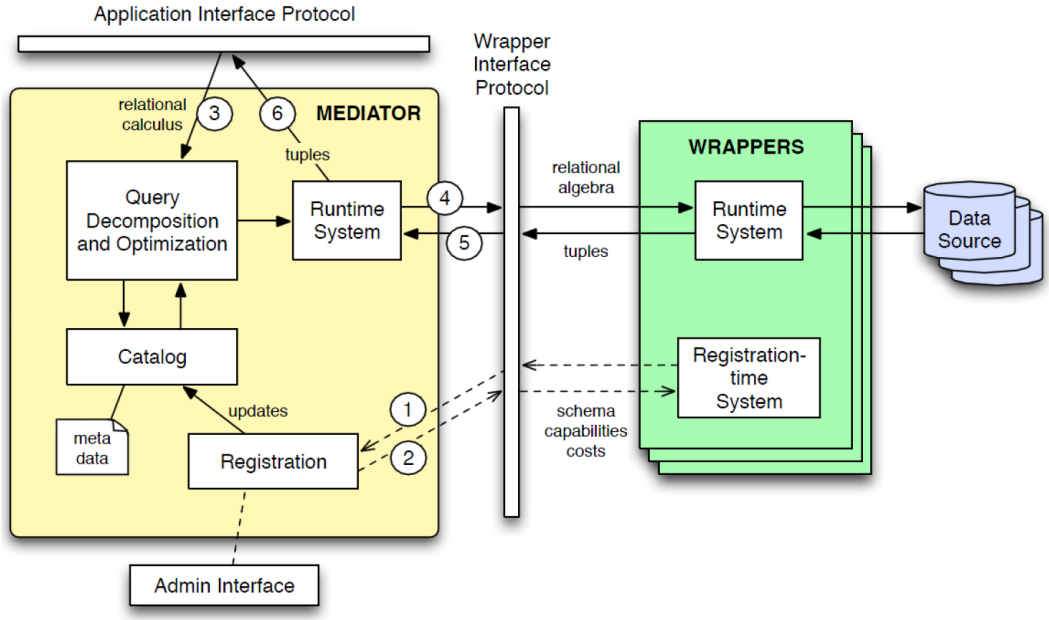

Рис. 1.4. Архітектура DISCO

На рисунку 1.4 числа показують порядок виконання роботи, починаючи з реєстрації нового адаптера джерела даних у посереднику DISCO, надання адаптером усієї необхідної інформації, такої, як експортована схема, описи можливостей і вартісних функцій та процес виконання запитів. На цьому малюнку також показано, що запити на мові глобальних запитів OQL обробляються на основі реляційного числення, однак, підзапити ґрунтуються на окремих алгебраїчних операціях. Наприклад, адаптер може надавати функцію доступу, але не операцію зв'язування в описі своїх можливостей. Результатами роботи адаптера завжди є кортежі [17].

Особливу увагу було приділено проблемі недоступності джерел даних. Система DISCO може надавати часткові результати, коли джерело даних стає недоступним під час обробки запитів. Дані, які не вдалося отримати, зберігаються у формі деякого запиту, який може бути виконаний пізніше для отримання повного результату. Так само, як і система Garlic, DISCO використовує стандарт ODMG-93 для опису глобальної моделі даних. [18] Однак. Замість створення власної мови запитів, глобальні запити в DISCO описуються на мові запитів ODMG, яка називається OQL (Object Query Language). Як спеціальна можливість, система DISCO показує метадані про зареєстровані адаптери, до яких можна спрямовувати запити за допомогою OQL, як до обгорнутих джерел даних.

Підтримка відображення моделей. У проєкті DISCO також вирішували завдання відображення моделей. Система DISCO надає три методи для інтеграції джерел даних, що мають локальні схеми, які не відповідають опису глобальної схеми. Однак пропонований підхід працює тільки між схемами ODMG. Таким чином, передбачається, що на першому етапі, адаптер уже надає ODMG схему для успадкованих джерел базової інформаційної системи.

У системі пропонуються такі три методи інтеграції [19]:

- завдання (визначення) підтипів для моделювання (subtyping for modeling) аналогічно підструктурам,
- відображення для моделювання подібних структур і подання для моделювання не однорідних структур. Метод визначення підтипів уже задано в стандарті ODMG, його добре використовують для об'єктів різного типу, але подібної структури.

Наприклад, якщо глобальна модель даних визначає тільки інтерфейс Person, а джерело даних має інтерфейс Student із схожою підструктурою, адаптер повинен визначити відношення підкласу для того, щоб включити клас Student як підклас глобального типу Person. Глобальний запит для отримання інформації про людей включатиме описи всіх студентів джерела даних. Визначення підтипів зазвичай використовується разом із відображенням.

Відображення в системі DISCO описується у вигляді частини визначення інтерфейсу. Існують два різні види відображень:

- відображення між локальними типами (наприклад, відношенням - relation) і глобальним типом;
- відображення між полем локального типу і полем глобального типу.

Наприклад, якщо поле, що визначає ім'я студента в типі Student, відрізняється від поля, що містить ім'я в типі Person, то може використовуватися відображення 1:1 (один до одного) для зв'язування полів з іменами. У системі DISCO використовуються тільки плоскі відображення. Немає можливості виконувати зв'язування між різними вкладеними структурами і немає способу вирішувати конфлікти схемної різнорідності. Також було тільки оголошено про підтримку функціональних відображень, але очевидно не було реалізовано.

Однак, у багатьох випадках різнорідних структур типів, поняття уявлень є дуже потужним підходом для узгодження типів. Насправді подання є виразами, що описують запити мовою OQL, які можуть генерувати необхідний віртуальний інтерфейс для узгодження з визначенням глобальних типів. Ба більше, подання може посилатися на інші типи. Якщо не враховувати питання продуктивності, то це надає потужний підхід, коли не достатньо більш простих відображень. Обробка запитів і вартісні функції.

Розробники системи DISCO могли отримати вигоду від раніше набутого досвіду під час виконання проєкту IRO-DB при створенні засобів обробки запитів і оптимізації. Архітектура проєкту IRO-DB вимагала, щоб кожен адаптер підтримував по-

вну мову OQL для відповіді на локальні запити. У результаті цього, для кожного адаптера потрібно було реалізувати процесор запитів на мові OQL (OQL query processor), що працює над вихідними джерелами інформації. [20]

Для локальних запитів, що надсилаються адаптерам, система DISCO використовує алгебраїчні оператори, аналогічні до тих, що використовувалися в системі Garlic, замість повної мови декларативних запитів. Такий підхід дає змогу реалізувати детальнішу модель вартості та поліпшити оптимізацію запитів. Оптимізатор системи DISCO ґрунтується на стандартних методах оптимізації реляційної бази даних. Усі плани мають глобальну частину і множинні локальні підплани, які безпосередньо надсилаються відповідним адаптерам. Для елемента PushDown системи Garlic є еквівалентний оператор у DISCO, який називається submit. Багато зусиль дослідників у проєкті DISCO було присвячено поліпшенню вартісної моделі [21].

1.6. Система InfoSleuth

Проєкт InfoSleuth (інформаційний шукач) був ініційований колись існуючою корпорацією "Microelectronics and Computer Technology Corporation (MCC)" у 1995, як продовження проєкту Carnot [14]. Проєкт Carnot був дуже раннім підходом (і, можливо, найпершим) до створення великомасштабної, багато-агентної програмної системи інтеграції інформації підприємств. На той час, запропоновані та реалізовані в цьому проєкті ідеї та концепції могли розглядатися як революційні. Як проєкт Carnot, так і проєкт InfoSleuth використовують онтології для підтримки семантичної інтеграції різнорідних джерел даних. У проєкті Carnot, використовується високо-рівнева онтологія Cyc [22] для відображення сутностей реального світу з так званими артикуляційними аксіомами на формально-визначені поняття онтології Cyc. У проєкті Carot було розроблено кілька прототипів застосунків для таких прикладних галузей, як управління робочими потоками, доступ до різнорідних баз даних, пошук знань у великих базах даних та інтегрований доступ до сховищ текстів і структурованих баз даних.

Проєкт InfoSleuth спонсорувався такими великими компаніями, як Andersen Consulting, Bellcore, Boeing, NCR/AT&T тощо, а також міністерством оборони США, що були зацікавлені в безвідмовному (fail-safe), масштабованому рішенні для інтеграції інформації, яку підтримують у великому розмаїтті географічно розподілених джерел. [23]

Ідея автономних програмних агентів, що діють від імені користувачів для розв'язання певної особливої задачі, прийшла зі спільноти дослідників у сфері штучного інтелекту та стала доволі популярною в 90-ті роки. Програмні агенти є значною мірою автономними комп'ютерними програмами (software program), які спілкуються по мережі (наприклад, мережі Інтернет) з іншими програмними агентами.

На Рисунку 1.5 показано узагальнену схему архітектури InfoSleuth. У мультиагентному середовищі, такому, як InfoSleuth, в основному є два види агентів: користувацькі агенти та інші, повністю незалежні агенти з дуже специфічними можливостями.

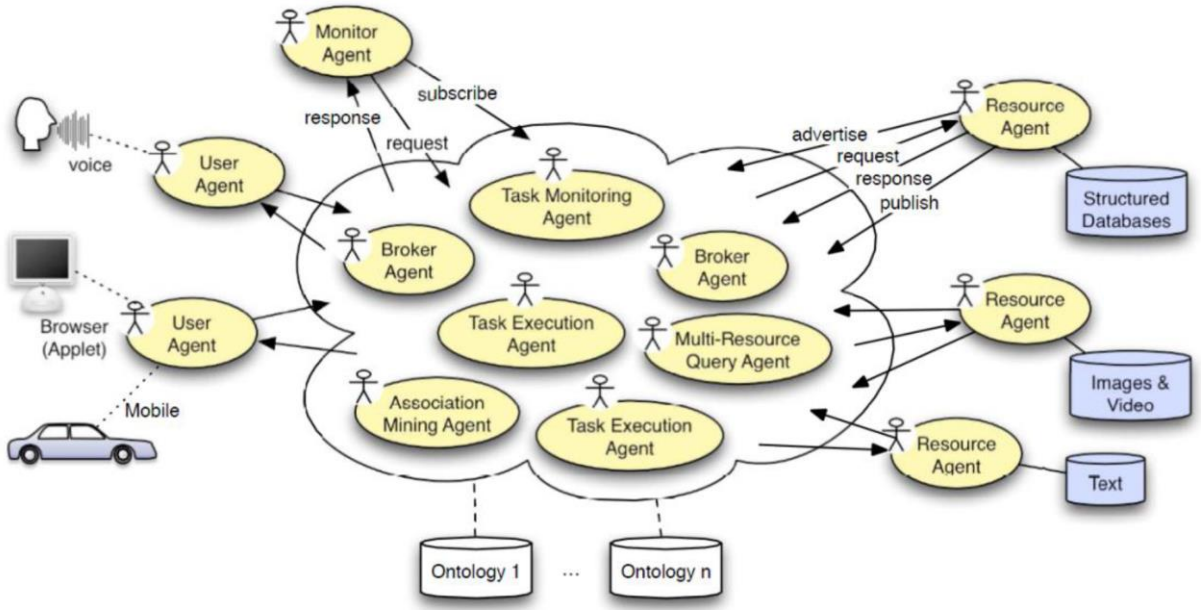

Рис. 1.5. Узагальнена схема системи InfoSleuth, що включає агентів, релевантних для інтеграції інформації

Наприклад, є такі агенти, як [24]:

- агенти-відстежувачі;

- зв'язувальні агенти;
- агенти, які виконують завдання;
- агенти, які відстежують виконання завдань;
- агенти, які аналізують взаємозв'язки;
- агенти, які виконують запити до набору ресурсів,;
- агенти ресурсів (агенти, що підтримують ресурс);
- та інші.

Користувацький агент використовується як інтелектуальний інтерфейс для людей - користувачів. Цю систему було реалізовано мовою Java для забезпечення незалежності від платформи. Призначений для користувача агент також може бути інтегрований у Web-сайти у вигляді Java-апплетів. Так само, як усі інші агенти, він сповіщає про своє існування і можливості агенту, що зв'язує, який діє як супер- в даній і передає повідомлення іншим агентам. Розміщення призначеного для користувача агента абсолютно не залежить від того, де перебуває сам користувач, що уможливлює використання цієї системи з будь-якого комп'ютера або пристрою, під'єднаного до мережі Інтернет. У той час, коли розроблялася система InfoSleuth, ще нічого не було відомо про таке пірингове (peer-to-peer) програмне забезпечення проміжного рівня, як, наприклад, JXTA, що було розроблено фірмою Sun у 2001 році [25]. Таким чином, систему InfoSleuth також можна розглядати як піонерський проєкт для пірингових мереж, які стали дуже популярними на початку 20 тисячоліття (і особливо після судового процесу над Napster, організованого RIAA (Recording Industry Association of America)).

Агенти ресурсів (resource agent, тобто адаптери) можуть розглядатися як воротарі інформаційних ресурсів (gatekeeper to an information source). Їхнім завданням є реагування (respond) на запити, відповіді на які можуть бути виведені з прикріплених до них джерел інформації. Такими джерелами можуть бути деякого виду бази даних, сховища мультимедіа або просто колекції текстових документів. Усі джерела інформації в системі InfoSleuth є повністю автономними. Зовні вони представляються на рівні релевантних їм семантичних понять, і у зв'язку з цим їхня локальна схема, формат і представлення є повністю незалежними.

Агент ресурсу є еквівалентом адаптера в архітектурі посередників-адаптерів. Він також підтримує відображення загальних онтологій предметних галузей на локальну схему і мову запитів, що є "рідним" для базового ресурсу. Агент ресурсу сповіщає про свої можливості: тобто, що він знає і якого виду відповіді він може формувати, і які запити він може приймати для цього ресурсу.

В архітектурі цієї системи не просто знайти еквівалент для посередника, тому що підкомпоненти посередників, такі, що є в системах Garlic, TSIMMIS або DISCO, розділені та виконуються різними агентами. Посередник, як його розуміють у системі Garlic, може складатися зі зв'язувального агента, агента виконання завдань, агента онтологій, агента виконання запитів до набору ресурсів і агента відстеження виконання завдань.[26] Система інтеграції SemWIQ.

Ще одним дослідженням у напрямі інтеграції великих обсягів даних є система SemWIQ, що базується на технологіях Семантичної Павутини.

Основними складовими елементами є клієнти, показані над посередником, що розташований у центрі, і кілька джерел даних, які показані в нижній частині рисунка 1.6.

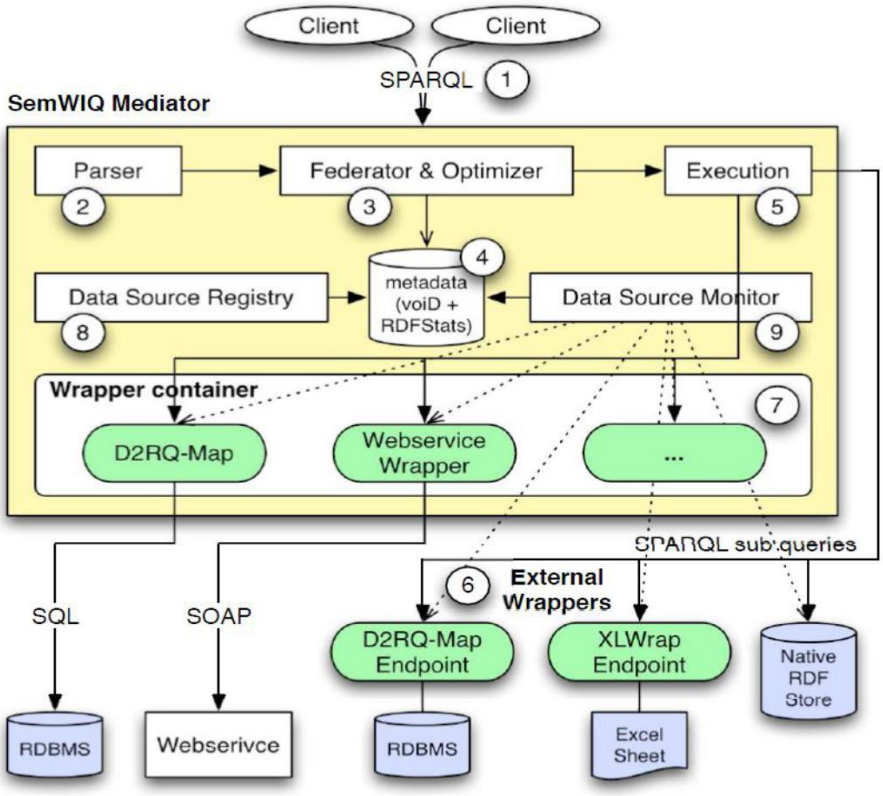

Рис. 1.6. Архітектура посередників-адаптерів у підході SemWIQ [13]

Користувачі системи SemWIQ виконують запити до посередника за допомогою клієнта (самостійна програма або web-додаток), що передають на опрацювання глобальні SPARQL запити (1). Глобальний SPARQL запит може містити не тільки будь-які термінологічні поняття, такі, як класи та властивості, але також і екземпляри понять, що існують у глобальному віртуальному графі. [27-28] Граматичний розбирач запитів (2) формує канонічний план запиту, який перетворюється та оптимізується інтегратором (3) на основі інформації, що міститься в каталозі метаданих (4).

У системі SemWIQ доступні два різні федератори. Розділений на частини та оптимізований план запиту остаточно обробляється підсистемою виконання запитів. Глобальний план запиту складається з декількох локальних підпланів, які далі об'єднуються високорівневими алгебраїчними операціями, що виконуються в посереднику. Будь-яка операція, яка може бути безпосередньо виконана адаптером віддалено, включається в план локального запиту. [29]

Віддалені адаптери розміщуються безпосередньо над віддаленими джерелами даних (6). Для джерел даних, які є повністю автономними, відповідні адаптери можуть поміщатися в спеціальний контейнер, що знаходиться в посереднику (7).

Журнал джерел даних (8) за реєстрацію та де-реєстрацію джерел даних. Джерело даних може бути зареєстроване та де-реєстроване шляхом надсилання HTTP POST запиту з відповідним URI кінцевої точки SPARQL. Монітор джерел даних (9) періодично виконує перевірку доступності зареєстрованих джерел даних, збирає метадані у форматі voiD та поточні статистичні дані RDFStats.

РОЗДІЛ 2
АНАЛІЗ ПІДХОДІВ ДЛЯ ОРГАНІЗАЦІЇ БАЗ ДАНИХ

Використання Semantic Web або онтологій регулярно розглядається і в науковому середовищі, і в середовищі практиків. Існує досить багато теоретичних і практичних робіт, які прямо або побічно зачіпають використання онтологій для впорядкування та стандартизації інформації, виходячи з вимог розробки RDF-графів, прийняту в Організацією W3C:

- наявність простої моделі даних;
- наявність формальної семантики і доказового логічного висновку;
- використання розширюваних словників на основі URI;
- використання синтаксису на основі XML;
- підтримка використання типів даних XML схеми;
- можливість будь-кому робити оголошення про будь-який ресурс [29].

2.1. Підхід Semantic Web

На сьогоднішній день Інтернет став персональним. Інтернет дедалі більше знає про його користувачів. Почасти, користувачі самі дають знання, публікуючи свої особисті дані в соціальних мережах, користуючись поширеними пошуковими системами, пройшовши авторизацію. З цього випливає, що вже завтра, вводячи в рядок пошуку "Де помити автомобіль недорого", користувач отримуватиме відповідь у вигляді найближчої автомийки та її місце розташування внаслідок чіткої відповіді на чітке запитання. Тепер не потрібно буде переходити за великою кількістю посилань із видачі пошукової системи різних пошукових систем, розчаровуючись у тому, що чергова відкрита вкладка - це черговий дорогий салон, просувний силами SEO фахівців. [30]

Це стосується різних сфер життя і діяльності людини - починаючи від побутових і закінчуючи більш глобальними. Наприклад, купівля автомобіля або квартири, пошук роботи та інші.

Ба більше, пошукова система зможе визначити, який саме автомобіль потрібен користувачеві на основі інформації про те, якими тест-драйвами він найбільше цікавиться та які автомобільні сайти відвідує, в якому районі та в якому ціновому діапазоні ви хочете знайти квартиру, чи не голодні ви, якій їжі надаєте перевагу тощо.

З розвитком семантичного вебу після збору певних даних про користувача технології дозволять скласти його соціально-демографічний портрет. Зібрані користувацькі дані комп'ютери розумітимуть уже як портрет особистості.

Багато в чому такій динаміці сприяє прагнення спростити сервіси і зробити спрощений доступ користувачів до контенту. Авторизація, що стала модною останнім часом, через соціальні мережі (Вконтакте, Facebook), спеціальні сервіси (OpenID, OAuth), коментування через віджети соціальних мереж [31].

2.2. Рамка опису ресурсу

RDF - це універсальний метод поділу знання на маленькі частини, відповідно до деяких правил, що враховують семантику (сенс) цих частин. Суть у тому, що такий метод має бути доволі простим, щоб за його допомогою можна було описати будь-який факт, і доволі структурованим, щоб представити факт у такій формі, у якій комп'ютерні додатки зможуть здійснювати корисні дії зі знаннями, вираженими у форматі RDF [32].

Стандарт RDF описує, як за допомогою об'єднання трьох таких сутностей можна констатувати деякий факт. Значення триплету '(Джон, Боб, відношення дружби)' полягає в тому, що Джон і Боб є друзями. Сукупність фактів у достатній кількості може бути деякою формою подання знань. Стандарти, засновані на RDF, включно з RDFS і OWL, доповнюють семантику RDF, роблячи можливим побудову логічних висновків на основі сукупності даних.

RDF забезпечує універсальний, гнучкий метод декомпозиції знань на маленькі частини, звані триплетами, за деякими правилами, що враховують семантику (сенс) цих частин. Основа методу в розбитті знань на частини і складанні того, що заведено

називати позначеним спрямованим графом. Кожна дуга такого графа являє собою деякий факт, відношення між двома сутностями.

Факт (або твердження), представлений у такий спосіб, складається з трьох частин: суб'єкт, предикат (аналогічний дієслову в природних мовах) і об'єкт. Суб'єкт представляється вузлом, з якого виходить дуга. Значення предиката визначається типом дуги (позначається за допомогою мітки дуги). Об'єкт - це вузол, до якого спрямована дуга.

Дані RDF подаються у вигляді RDF Schema (RDFS). RDFS являє собою словник для опису RDF триплетів. Використання RDFS уможливлює обробляти дані в RDF графі.

Багато частин словника RDFS увійшли в OWL. OWL теж являє собою словник. Його основною функцією є описування ресурсів, що належать до веб-сторінок.

2.3. Мова запитів SPARQL

SPARQL є мовою запитів до RDF документів. Вона прийнята як стандарт Консорціумом Всесвітньої павутини. Створення точок доступу SPARQL рекомендується при створенні та публікації веб-сторінок.

2.4. Реляційні бази даних

Реляційна база даних - це сукупність взаємопов'язаних таблиць, кожна з яких містить інформацію про об'єкти певного типу [15]. Рядок таблиці містить дані про один об'єкт (наприклад, товар, клієнта), а стовпці таблиці описують різні характеристики цих об'єктів - атрибутів (наприклад, найменування, код товару, відомості про клієнта). Записи, тобто рядки таблиці, мають однакову структуру - вони складаються з полів, що зберігають атрибути об'єкта. Кожне поле, тобто стовпець, описує тільки одну характеристику об'єкта і має чітко визначений тип даних. Усі записи мають одні й ті самі поля, тільки в них відображаються різні інформаційні властивості об'єкта. [33]

2.5. Пропонований підхід до вирішення досліджуваної задачі

Запропоновано використовувати такі методи та моделі для інтеграції реляційних баз даних:

- Перетворення реляційних баз даних на RDF.
- Створення алгоритму інтеграції онтологій з RDF графів, утворених баз даних.
- Розробка та реалізація серверної частини веб-сервісу.
- Розробка та реалізація клієнтської частини веб-сервісу.

…

РОЗДІЛ 3
МЕТОД ІНТЕГРАЦІЇ РЕЛЯЦІЙНИХ БАЗ ДАНИХ ІЗ ВИКОРИСТАННЯМ ПЛАТФОРМИ D2RQ

3.1. Аналіз рішення

Для реалізації механізму інтеграції реляційних баз даних необхідно детально розробити архітектуру додатка або сервісу. Насамперед необхідно визначити функції, які виконуватиме програмний продукт, виявити необхідні вимоги.

Розроблення архітектури програмного забезпечення - це вид діяльності на етапі проєктування ПЗ, пов'язаний із визначенням основних обмежень, що накладаються на проєктування системи, як-от вибір парадигми програмування, архітектурних стилів, стандартів розроблення ПЗ. Вони ґрунтуються на використанні компонентів, принципів проєктування та обмеження, що накладаються державним законодавством. Детальне проєктування, тобто розробка тактики, - це діяльність, пов'язана з визначенням локальних обмежень проєкту, як-от шаблони проєктування, архітектурні моделі, ідіоми програмування та рефакторингу [34].

3.2. Архітектурний стиль програми

Ґрунтуючись на функціональних вимогах і спрощеній моделі системи, можна зробити висновок: інформаційна система має бути виконана у вигляді веб-сервісу.

Це пояснюється тим, що веб-додатки на даний момент є найбільш крос-браузерними додатками. Для їх використання необхідний лише веб-браузер. На Рисунку 3.1 можна побачити найпоширеніші операційні системи для персональних комп'ютерів і мобільних пристроїв, а також найпоширеніші браузери. [21, 34]

Рис. 3.1. Діаграма використання ОС і браузерів

Найпопулярніші браузери, а саме FireFox і Google Chrome, мають реалізацію на найуживаніших платформах: сімейства Windows, Linux і MacOS, Android і iOS. Отже, реалізація поставленого завдання з інтеграції у вигляді веб-додатка враховує одразу кілька функціональних вимог: доступність широкому колу користувачів, відсутність необхідності встановлювати додаткове програмне забезпечення.

З огляду на основну функцію інформаційної системи можна зробити висновок, що найкращим архітектурним стилем для її реалізації є сервісо-орієнтована архітектура. Вона передбачає наявність модулів, які слабко пов'язані між собою і легко замінні. Взаємодія реалізується за допомогою різних інтерфейсів і протоколів.

3.3. Опис моделі системи

Наступним етапом проєктування є створення моделі системи. Цей етап необхідний для визначення необхідного програмного забезпечення: мови програмування, шаблонів проєктування, програмних бібліотек, середовища розробки та платформи для майбутньої публікації. Проста модель інформаційної системи має зачіпати лише основні пункти її роботи. Також необхідно врахувати інформаційні потоки, які передають необхідні дані між різними учасниками інформаційної системи. Для початку варто виділити основні пункти роботи системи [35]:

1. Отримання доступу до реляційних баз даних.

2. Подання реляційних баз даних у вигляді RDF графів.
3. Аналіз отриманих RDF -графів.
4. Інтеграція даних.
5. Створення RDF графа на основі інтеграції.
6. Виведення результату.

На рисунку 3.2 подано приклад моделі інформаційної системи. За допомогою клієнтської частини додатка користувач може ініціювати запит до сервера. Сервер запитує дані з реляційних баз даних. Дані повертаються через посередника. Посередник перетворює реляційні бази даних на RDF графи. На сервері відбувається інтеграція отриманої інформації. І користувач отримує відповідь.

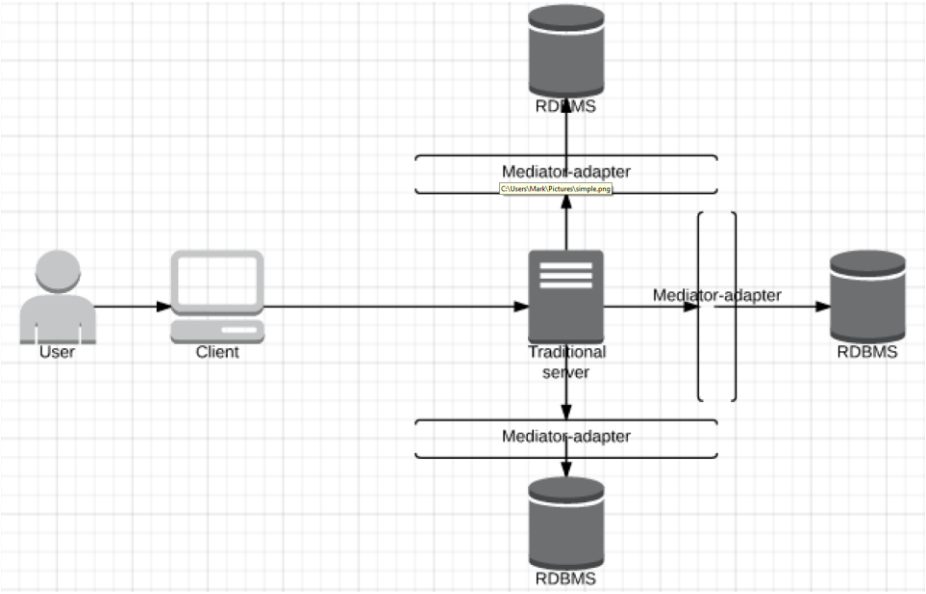

Рис. 3.2. Приклад модель інтеграції реляційних БД інформаційної системи

3.4. Вибір програмних засобів

Для реалізації інформаційної системи у вигляді веб-додатка необхідно вибрати програмні засоби, які реалізують інтеграцію реляційних баз даних, використовуючи Semantic Web, з можливістю розгортання програми в мережі Інтернет.

Посередник-адаптер. Однією з основних задач є перетворення реляційної бази даних на RDF - граф. Його буде вирішено за допомогою посередника-адаптера. Таким медіатором може виступати платформа D2RQ [36].

Платформа D2RQ. Для створення адаптерів-посередників у розроблюваній інформаційній системі, як і в підході SemWIQ, можна використовувати платформу D2RQ.

Платформа D2RQ - це інформаційна система для надання доступу до реляційних база даних як віртуальних RDF графів, що доступні лише для читання. Зокрема, доступ до RDF вмісту реляційних баз даних відбувається без їхнього перетворення на RDF сховища.

Використовуючи платформи D2RQ можливо [37]:

- здійснювати запити до реляційної бази даних, використовуючи мову запитів SPARQL;
- отримувати доступ до реляційних баз даних через Web-мережу як набір Пов'язаних даних;
- створювати свої RDF дампи для подальшого завантаження їх у RDF сховища;
- працювати з інформацією з реляційних баз даних, використовуючи різні програмні бібліотеки для роботи з RDF.

Склад D2RQ-платформи:

- generate-mapping;
- d2r-сервер;
- d2r-запит;
- dump-rdf;
- D2RQ+Jena API.

На рисунку 3.3 показано архітектуру платформи D2RQ

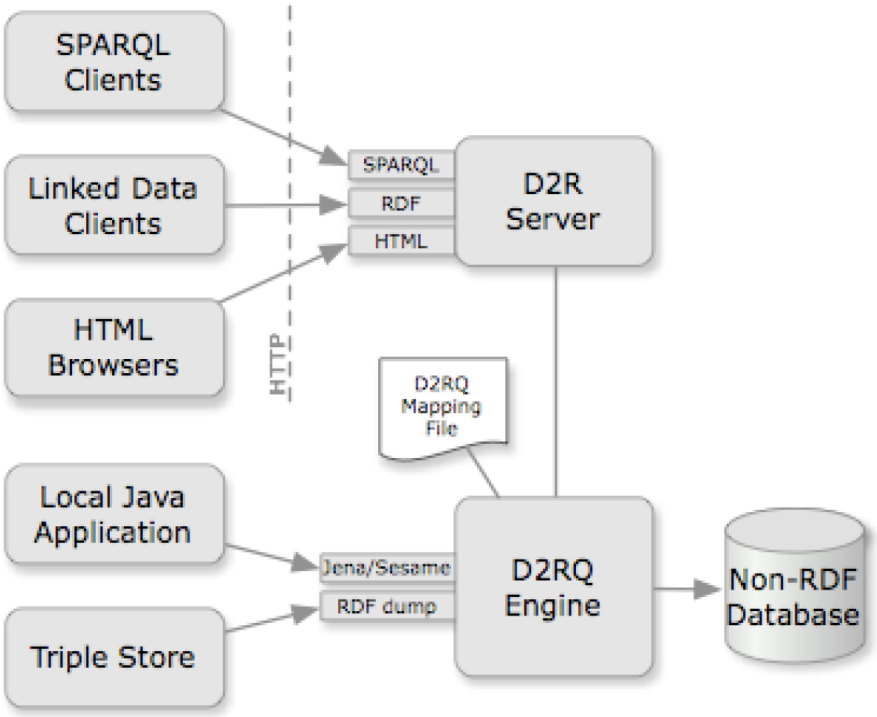

Рис. 3.3. Архітектура платформи D2RQ

Формування відображення. Інструмент generate-mapping (формування відображення) створює файл зіставлення D2RQ шляхом аналізу схеми наявної бази даних. Цей файл опису відображення, називається default mapping, він відображає кожну таблицю як RDFS клас, що базується на імені таблиці, і відображає кожний стовпчик у власності, що базується на імені стовпця. Цей mapping файл може бути використаний як є або може бути змінений. Сервер D2R

D2R-сервер є інструментом для публікації реляційних баз даних на технології Semantic Web. Це дає змогу RDF і HTML-браузерам здійснювати навігацію змісту бази даних, а також дає змогу запитувати базу даних за допомогою мови запитів SPARQL [33].

Дані про Semantic Web моделюються і представляються в RDF. D2R сервер використовує файл відображення (mapping файл) для представлення D2RQ вмісту бази даних, і дає доступ до даних RDF для перегляду і пошуку - дві основні парадигми доступу до Semantic Web. SQL запити з Web переформуються за допомогою generate-mapping. Це дає змогу публікувати RDF від великих робочих баз даних і прибирає необхідність обробки даних у виділеному RDF-триплеті.

Можливості D2R-сервера:

- Зручний перегляд вмісту бази даних.

Простий веб-інтерфейс дає змогу здійснювати пошук у вмісті в базах даних і надає користувачам RDF-даних у зручному форматі під час попереднього перегляду.

- Складання зручних URI (Uniform Resource Identifier).

На основі принципів Linked Data, D2R сервер призначає URI для кожного об'єкта, описаного в базі даних, тобто, опис RDF можна відновити через власника URI. Semantic Web браузери, наприклад, Marbles або LinkSailor можуть використовувати URI для навігації по Semantic Web.

- Висока узгодженість

D2R-server створює однакові URI, що полегшує роботу з ними, використовуючи інші платформи. Також у D2R-sever реалізовано загальноприйняті інтерфейси, наприклад HTTP.

- Підтримка SPARQL

Дозволяє запитувати дані через SPARQL запити через мову запитів - SPARQL 1.1. Також має найпростіший вбудований текстовий редактор SPARQL запитів.

- Робота з URI

D2R-server дає змогу не тільки створювати URI, а й змінювати наявні, конфігурувати файли з триплетами, які в них зберігаються.

Компонент D2R-query. Компонент D2R-query дозволяє виконувати запити SPARQL через D2RQ-карту реляційної бази даних з командного рядка. Це може бути зроблено з використанням або без mapping файлу. Якщо mapping файл задано, то інструмент запитуватиме віртуальний RDF граф, визначений mapping файлом. Якщо файл відображення не вказано, то інструмент використовуватиме відображення за замовчуванням [35].

Компонент Dump-RDF. Компонент Dump-RDF створює D2RQ файл із вмістом усієї бази даних в одному файлі RDF. Це може бути зроблено з або без файлу відображення D2RQ. Якщо файл відображення задано, то інструмент використовуватиме його, щоб перевести інформацію баз даних RDF. Якщо файл відображення не вказано, то інструмент використовуватиме відображення за замовчуванням.

Точка доступу D2QR + Jena API. Jena - це Java фреймворк для створення додатків, що працюють із Semantic Web. D2RQ може використовуватися як компонент для отримання доступу до RDF.

D2RQ дає змогу Jena витягувати дані на різних рівнях:

1. Jena Model API.
2. Jena Graph API.
3. SPARQL запити до RDF.

Серверна та клієнтська частини. Для розв'язання задачі з реалізації алгоритму інтеграції та серверної частини додатка необхідно використати потужний інструмент, що передбачає можливість роботи з RDF-графами, реалізує багатопоточність, а також забезпечить функціонування веб-сервісу. Таким інструментом є мова програмування Python. [38]

Мова програмування Python. Python - мова програмування високого рівня. Вона призначена для виконання завдань загального призначення. Python орієнтована на високу швидкість написання коду, а також зручність його читання. Стандартна бібліотека Python містить великий і різноманітний набір методів, хоча ядро Python виконує тільки основні функції, які характерні для мов програмування високого рівня. Python - мультипарадигмна мова програмування. Вона підтримує:

- структурне програмування;
- об'єктно-орієнтоване програмування;
- функціональне програмуванням;
- імперативне програмуванням;
- аспектно-орієнтоване програмування.

Також Python має динамічну типізацію, автоматичне керування пам'яттю, обробку помилок, повну інтроспекцію та можливість створення багатопотокових програм. Його широта використання та простота вивчення отримали широкий відгук не лише в професійному середовищі, а й у науковому. Завдяки цьому кількість бібліотек і фреймворків для Python постійно поповнюється. Серед них можна виділити бібліотеки, які знайшли застосування в науковому середовищі:

1. SciPy - бібліотека, що містить пакети для інтегрування, генетичних алгоритмів, розв'язування диференціальних рівнянь та інших задач, які розв'язуються в науці або під час інженерного розроблення.

2. Matplotlob - бібліотека для візуалізації 2D і 3D графіки.

3. NumPy - розширення, яке додає підтримку багатовимірних масивів і матриць, а також багато математичних функцій, необхідних для роботи з ними.

На підставі широкого застосування Python у науковому середовищі, а також наявності в ньому великої кількості необхідних програмних засобів у вигляді бібліотек і фреймворків, Python ляже в основу проєктованої програмної системи [36].

Фреймворк Django. Для створення веб-додатка буде використовуватися Django Python Framework. Планування та реалізація Web-сайтів завжди супроводжується великими витратами зусиль. Django являє собою один із найкращих на на сьогоднішній день фреймворків, який дає змогу швидко вести розробку високопродуктивних і повнофункціональних сайтів. За допомогою django легко вистоюються масштабовані і легко розширювані додатки для Web з дизайном будь-якого ступеня складності.

Абстрагуючись від низькорівневого процесу Web-розробки, django дає змогу розробникам швидко створювати динамічні Web-сайти, що базуються на базах даних. Однією з основних переваг django є переносимість створених на її основі продуктів через переносимість їхнього базису - мови високого рівня Python.

Django містить Model View Controller (MVC) - шаблон проєктування, що дає змогу розділити загальну архітектуру на окремі частини. При цьому керуюча логіка розділена на три окремі компоненти так, що модифікація одного з них має мінімальний вплив на інші частини. До таких компонентів відносять дані, що розділяються, логіку і шари візуалізації. У загальному випадку така концепція дає змогу розділити розроблення інформаційного наповнення на рівні бази даних і розроблення Web-сторінок.

Django базується на класі Python django.db.models.Model, який задає дані моделі так, щоб вони були придатні до використання на Web-сайтах. Ці дані визначаються відповідними атрибутами об'єктів, які зберігаються в базі даних у процесі роботи.

При створенні сайту створюється підклас класу Model і додається поле членів у клас для завдання специфічних даних.

Інтерфейс моделі django забезпечує багатоплановий вибір з усіх наявних типів моделей для відбору найбільш підходящого в даній ситуації. Обрана модель проєкту синхронізується з працюючою базою даних, де вона зберігає свої дані в таблицях. Django взагалі забезпечує хороший інтерфейс до баз даних, що дає змогу отримувати надійний доступ до інформації з шарів візуалізації та шаблонів.

Зміна відображення вмісту залежно від прийнятого URL-запиту є багатоступеневим процесом. Коли django-сервер отримує URL-запит, він парсить його і, використовуючи попередні установки шаблонів, визначає, яку ділянку коду Python буде виконуватися для необхідного відображення.

Парсер шаблонів у django дає змогу самостійно налаштовувати свої шаблони, які використовують функції відображення Web-сторінок під час побудови відповіді на URL-запити. Це дає змогу розробникам Python сфокусуватися на створенні даних, які будуть відображатися, а програмістам HTML - сфокусуватися на дизайні Web-сторінок [37].

Можливості фреймворка:

- ORM, API доступу до БД із підтримкою транзакцій;
- вбудований інтерфейс адміністратора, з вже наявними перекладами багатьма мовами;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів із тегами та успадкуванням;
- система кешування;
- інтернаціоналізація;
- підключається архітектура додатків, які можна встановлювати на будь-які Django-сайти;
- "generic views" - шаблони функцій контролерів;
- авторизація та автентифікація, підключення зовнішніх модулів автентифікації: LDAP, OpenID та ін.;

- система фільтрів ("middleware") для побудови додаткових оброблювачів запитів, як-от, наприклад, включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій;
- бібліотека для роботи з формами (успадкування, побудова форм за наявною моделлю БД);
- вбудована автоматична документація щодо тегів шаблонів і моделей даних, доступна через адміністративний застосунок [39] Також Django може реалізувати або взаємодіяти з HTML формами та JavaScript. Ці технології є стандартом для створення клієнтських частин веб-додатків.

Програмна бібліотека RDFLib. Оскільки в реалізації додатка з інтеграції реляційних даних мова програмування Java використовуватися не буде, необхідно знайти програмну бібліотеку для вилучення RDF графів під час виконання програми. Для мови Python існує досить багато бібліотек для роботи з RDF. Для впровадження в проєкт було обрано бібліотеку RDFLib, оскільки:

- бібліотека містить необхідні парсери та серіалізатори для RDF і метаданих;
- бібліотека є інтерфейсом для роботи з RDF-графами;
- бібліотека містить вбудоване сховище RDF-графів;
- бібліотека підтримує мову запитів SPARQL;
- вона є вільно розповсюджуваною і зберігається на ресурсі GitHub;

Платформа для розгортання. Для публікації, розгортання та використання веб-додатка необхідно розгорнути проєкт на будь-якому з доступних джерел, як-от хмарні сервіси (Heroku, Google App Engine) або окремому сервері. Оскільки мовою програмування є Python у зв'язці з Django Framework, то варто вибрати сервіс, який задовольняє вимогам до розробки всієї системи інтеграції. За функціональністю результати у використанні сторонніх сервісів або налаштування власного сервера нічим не відрізняються. Але з економічної точки зору різниця між щомісячною оплатою доменного імені та внесення щомісячних платежів на сторонні ресурси є принциповою. Найбільш підходящим варіантом є використання Ubuntu Server 22.04 LTS у зв'язці з

пакетами nginx для швидкої відмовостійкої роботи та uwsgi для розгортання додатка, написаного на Django.

Ubuntu Server - це вільно розповсюджувана операційна система, заснована на Debian. Ubuntu Server поставляється з набором програм необхідних для роботи сервера або робочої станції. Також у ній є розширена локалізація і підтримка основних процесорних архітектур.

3.5. Вимоги до програмної реалізації

Призначення програми. Розроблюваний веб-додаток слугує для інтеграції реляційних баз-даних, використовуючи підхід Semantic Web.

Область застосування. Цей програмний продукт є сервісом, що дає змогу перетворити реляційні бази даних, у яких можуть бути синонімічні поля, і на підставі цього отримати RDF-граф, що містить безліч триплетів, які характеризують інтегровані реляційні бази даних як одне ціле.

Для проєктування архітектури додатка необхідно визначити список необхідних вимог до програмного продукту.

Функціональні вимоги. Функціональними вимогами є набір тверджень щодо поведінки інформаційної системи, обмежень.

Ґрунтуючись на необхідності створення дешевої у виконанні та експлуатації системи, а також її доступності для кінцевого користувача, можна виділити такі функціональні вимоги:

- Інформаційна система повинна реалізовувати інтеграцію реляційних баз даних, використовуючи Semantic Web.
- Інформаційна система повинна бути максимально доступна широкому колу користувачів.
- Для експлуатації інформаційної системи повинен бути необхідний лише базовий набір програмного забезпечення, який постачають розробники операційних систем.

- Інформаційна система має бути автономною, тобто зміна сторонніх додатків, що використовуються в ній, не повинна позначатися на працездатності системи.
- Інформаційна система повинна інтегрувати віддалені джерела різних реляційних баз даних.

Вимоги до надійності. Робота системи має гарантувати середній аптайм - 99 % на рік. Для цього система має бути розгорнута на хмарному сервісі, що надає платформу як послугу, або на сервері організації-клієнта під управлінням останньої стабільної версії Windows Server або серверної версії операційної системи Linux.

У разі введення некоректної інформації програмна система має видати попереджувальне повідомлення про некоректність введених даних і запропонувати повторити введення.

Відновлення системи після критичних помилок, що призводять до непрацездатності системи, має відбуватися протягом доби після моменту виникнення помилки.

Вимоги до ергономіки та технічної естетики. Взаємодія користувача і програми здійснюється за допомогою графічного інтерфейсу клієнтської програмної системи - браузера. Програма повинна мати елементи навігації. Програма має бути виконана в єдиному стилі, мати однакове розташування основних елементів навігації. Програма повинна підтримувати гарячі клавіші підтвердження введення - Enter і закриття підказок і спливаючих вікон - Escape.

При введенні некоректних даних, повинні видаватися повідомлення на основною мовою програмної системи про неправильне введення даних і пропозицію ввести дані заново.

3.6. Проєкт системи

Діаграма проєктних класів. Проєктні класи - це класи, опис яких настільки повний, що їх можна реалізувати. Опис формується шляхом уточнення класів аналізу, що включає в себе додавання деталей реалізації. На діаграмі 3.4 наведено діаграму проєктних класів, що реалізуються в системі.

На Рисунку 3.4 також показано, що інформаційна система складається з двох пакетів: WebInterface і Seweb.

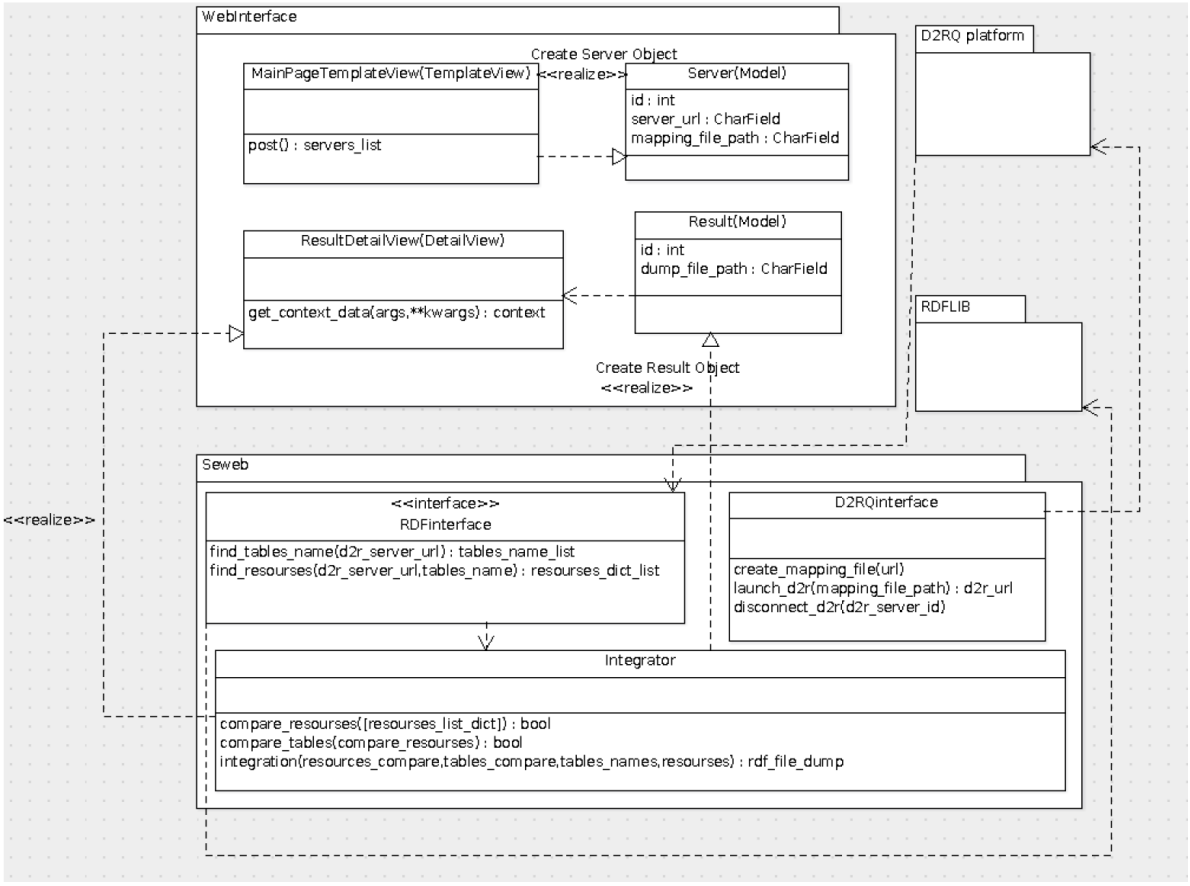

Рис. 3.4. Діаграма класів системи інтеграції SeWEB

Пакет WebInterface містить класи MainPageTemplateView і ResultDetailView, що успадковуються від Class Based View класів Django та слугують для опрацювання вмісту сторінки або опрацювання запитів із HTML-форм, які надсилаються в HTTP-запитах.

Пакет Seweb призначений для роботи з RDF графами: отриманням, порівнянням і створенням нових. Він складається з класів: D2RQinterface, RDFinterface та Integrator.

Клас D2RQinterface організовує роботу і розгортання локальних D2R серверів в окремих потоках, використовуючи стандартну бібліотеку Python для роботи з потоками - GIL.

Клас RDFinterface організовує містить методи для отримання RDF графів, що знаходяться на D2R-серверах, і даних з них за допомогою RDFLib

Клас Integrator містить методи, що реалізують перевірки на збіг предикатів і об'єктів однакового суб'єкта RDF графів із різних D2R-серверів. На підставі цього він може створити новий RDF-граф і записати його в базу даних проєктованого застосунку, або ж повідомити системі, що збігів немає.

Також варто відзначити наявність бібліотеки RDF і D2RQ-платформи в інформаційній системі. Їхня наявність пояснює взаємодію класів між собою.

Діаграма варіантів використання. Діаграма варіантів (рисунок 3.5) використання показує можливості користувача в системі і ґрунтується на функціональних вимогах до розроблюваної системи.

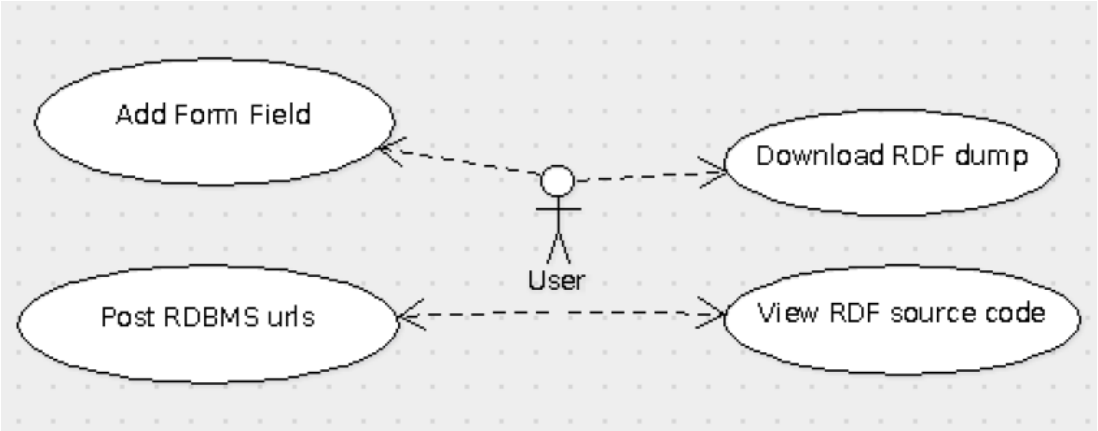

Рис. 3.5. Діаграма варіантів використання

Діаграми послідовностей для операцій проєктних класів. Діаграма послідовності (англ. sequence diagram) - діаграма, на якій показана взаємодія об'єктів (обмін між ними сигналами та повідомленнями), впорядкована за часом, з відображенням тривалості опрацювання та послідовності їхніх проявів [20].

Невід'ємною частиною об'єкта на діаграмі послідовності є лінія життя об'єкта. Лінія життя показує час, протягом якого об'єкт існує в Системі. Періоди активності об'єкта в момент взаємодії показуються за допомогою фокуса управління. Тимчасова шкала на діаграмі спрямована зверху вниз [40].

Під час виконання проєкту було спроєктовано діаграми послідовностей для кількох функцій. На діаграмі 3.6 зображено діаграму послідовності для методу launch_d2r.

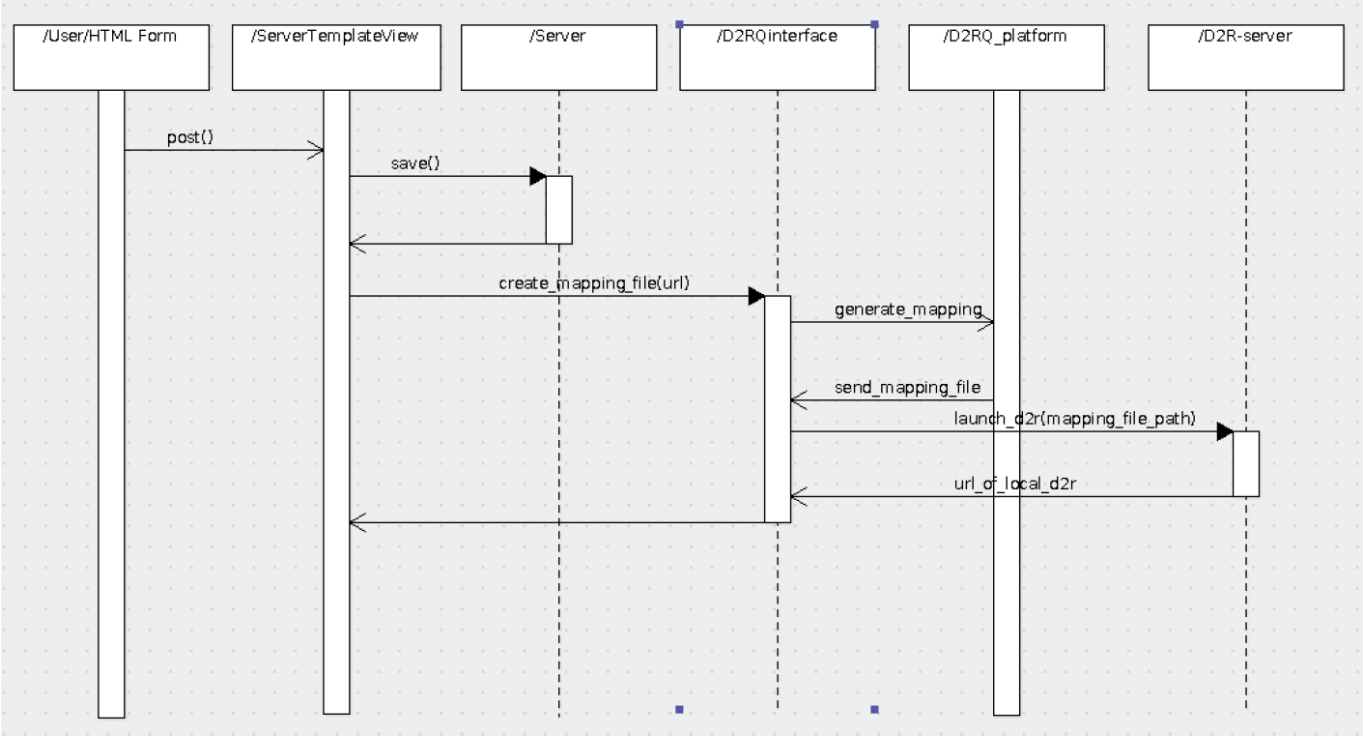

Рис. 3.6. Діаграма послідовності для методу launch_d2r(url)

Діаграма описує надсилання користувачем(User) URL віддалених реляційних баз даних. Масив з URL передається на обробку в backend веб-додатку за допомогою POST запиту за протоколом HTTP. Масив обробляється в тілі методу post() класу ServerTemplateView. У ньому виклик функції save() з класу Server. Клас Server успадкований від класу Model, який є вбудованим класом Django і містить необхідні методи для роботи з базою даних. Після збереження викликається метод create_mapping_file() з параметром url з класу D2RQinterface. З цього класу відбуваються запити на створення mapping-file і створення потоку, в якому локально буде розгорнуто D2R server.

На діаграмі 3.7 можна побачити опис роботи методу find_tables_name(), який викликається з головного потоку, під час роботи локально розгорнутих серверів D2R.

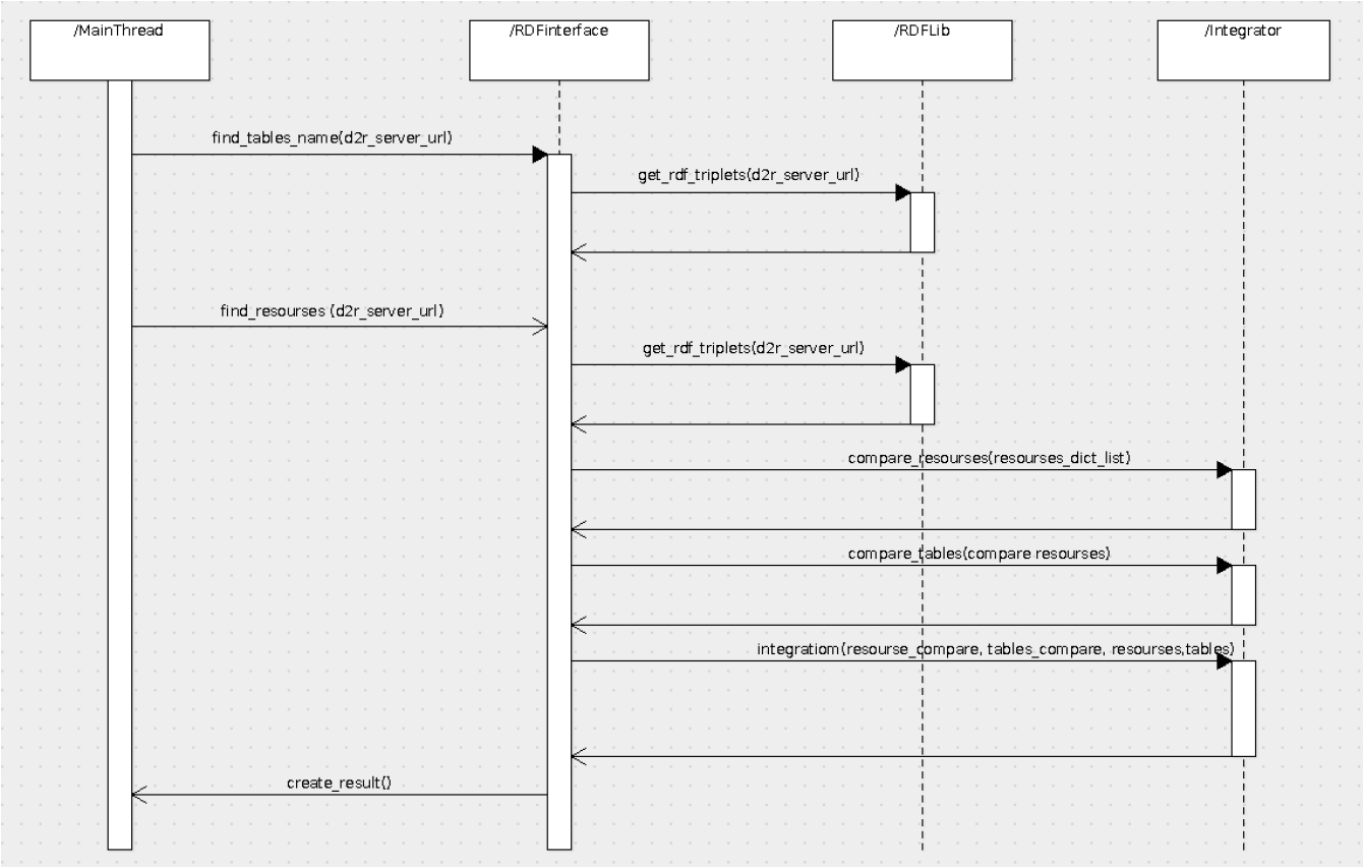

Рис. 3.7. Діаграма послідовності для методу launch_d2r(url)

Після виклику цього методу опрацювання програми починається в класі. RDFinterface, який містить функції для знаходження назви таблиць реляційної бази даних, а також ресурсів. По черзі метод надсилає запити в бібліотеку RDF для отримання повної інформації про дані, які його цікавлять. Після отримання списку словників, що містять необхідні значення, починається взаємодія з класом Integrator. У цьому класі спочатку відбуваються порівняння між списками, отриманими з різних D2R-серверів, і якщо порівняння повертають True, викликається клас метод integration, в якому формується новий RDF-граф, на основі порівняних даних.

3.7. Архітектура веб-сервісу

Архітектура веб-сервісу містить у собі:

- Платформу D2RQ - як посередник адаптер.
- Python/Django Framework - реалізація серверна частина.

- RDFLib - реалізація інтерфейсу для обробки RDF-графів.
- HTML/JavaScript - реалізація клієнтської частини.
- Ubuntu Server - платформа для розгортання.

Архітектура додатка показана на схемі 3.8.

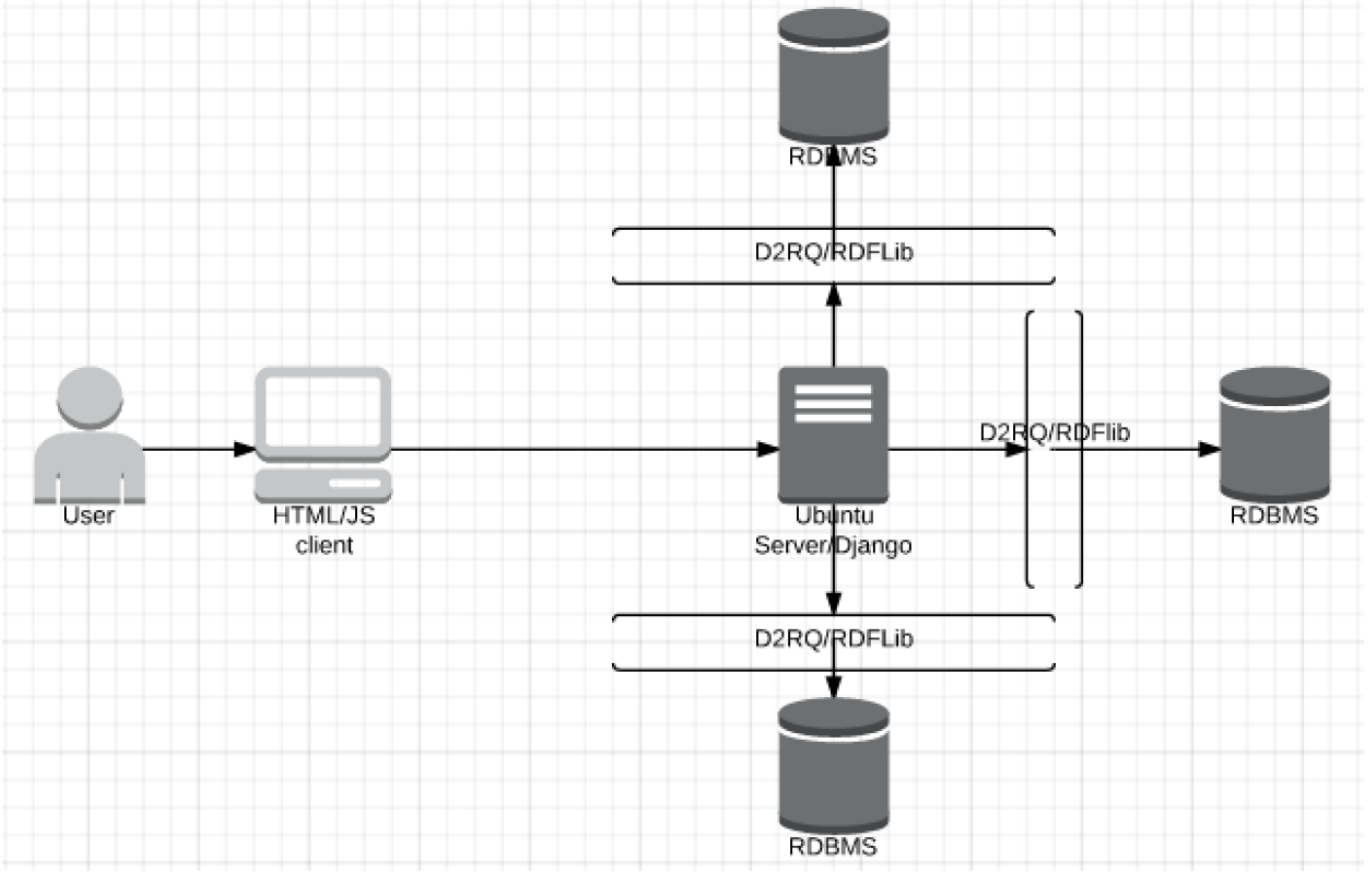

Рис. 3.8 . Архітектура веб-сервісу для інтеграції реляційних баз даних

На цьому малюнку показано такі кроки інтеграції:

1. Користувач надсилає URL реляційних баз даних через HTML форми методом POST Сервер (Django) обробляє запит і надсилає URL для створення mapping файлів.
2. На підставі mapping файлів локально запускаються D2R – сервери.
3. RDFlib отримує дані з D2R-серверів і класифікує їх на імена таблиць і сутності.
4. Python проводить інтеграцію реляційних баз даних.
5. Результат інтеграції обробляється в Django для виведення користувачеві.

6. Результат представлений у вигляді RDF і виводиться на HTML-сторінку.

Програмна реалізація системи. У результаті було розроблено веб-сервіс, що реалізує інтеграцію реляційних баз даних. До складу веб-сервісу входять:

- Python+Django+RDFlib - як серверна частина інформаційної системи.
- D2RQ - як медіатор-адаптер для реляційних баз даних.
- HTML+JavaScript+CSS - як клієнтська частина інформаційної системи.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ІНТЕГРАЦІЇ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

У даному розділі кваліфікаційної роботи описано процес розробки веб-сервісу, що реалізує інтеграцію реляційних баз даних. До складу веб-сервісу входять:

- Python+Django+RDFlib - як серверна частина інформаційної системи.
- D2RQ - як медіатор-адаптер для реляційних баз даних.
- HTML+JavaScript+CSS - як клієнтська частина інформаційної системи.

4.1. Реалізація серверної частини веб-сервісу

Основний функціонал розробленої системи міститься в python-пакеті Seweb. Пакет Seweb складається з трьох класів: Integrator, D2RQinterface, RDFinterface

1. Integrator призначений для порівняння суб'єктів RDF-триплетів і складання RDF-дампа, у якому містяться триплети інтегрованого графа у форматі, доступному для додавання в RDF-сховище.
2. D2RQinterface призначений для динамічного створення і завершення.
3. RDFinterface призначений для роботи з RDF графами. Він тісно пов'язаний з використанням бібліотеки RDFlib.

У пакеті Webinterface знаходяться класи і методи для обробки HTTP запитів, а також організацію введення-виведення інформації з HTML сторінок.

4.2. Приклад налаштування платформи D2RQ Вхідні параметри

Порядок роботи з D2RQ платформою полягає в послідовному використанні доступних інструментів платформи. Вхідним параметром є реляційна база даних. Для прикладу, зображеного на малюнку 4.1, буде використовуватися СУБД MySQL.

mysql> SELECT * FROM student_list;

fullname	date_of_birth	group_number	id
Andrii Stepaniuk	16.06.2001	TK-247M	1
Liana Vasylenko	10.07.2001	TK-247M	2
Vladyslav Pulnyi	24.11.2000	TK-247M	3
Valentyn Khivrykh	11.02.2001	TK-247M	4

Рис. 4.1. Приклад таблиці з описами студентів у базі даних СУБД MySQL

Generate – mapping. Для розгортання D2R сервера необхідно створити mapping файл реляційної бази даних. Для цього використовується інструмент D2RQ generate-mapping. У вхідних необхідних параметрах він приймає JDBC Driver із зазначенням місця зберігання бази даних, логін і пароль для роботи з нею. У вихідних параметрах слід вказати ім'я створюваного файлу відображення. На малюнку 4.2. показано варіант створення mapping-файлу

```
computer ~/java_workspace/Jena/d2rq $ ./generate-mapping -u root -p Doshira
o osu_mapping.ttl  jdbc:mysql://localhost/osu
```

Рис. 4.2. Створення mapping файлу

У вихідному файлі кожну таблицю бази даних представлено як окремий RDFS клас, який заснований на імені таблиці, а його властивості - це колони в таблиці.

Таблиця 4.1

Приклад змісту mapping файлу

```
@prefix map: <#> .
@prefix db: <> .
@prefix vocab: <vocab/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .
map:database a d2rq:Database;
    d2rq:jdbcDriver "com.mysql.jdbc.Driver";
    d2rq:jdbcDSN "jdbc:mysql://localhost/osu";
    d2rq:username "root";
    d2rq:password "Doshirak24";
jdbc:autoReconnect "true";
jdbc:zeroDateTimeBehavior "convertToNull";
# Table student_list
map:student_list a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "student_list/@@student_list.id@@";
    d2rq:class vocab:student_list;
    d2rq:classDefinitionLabel "student_list";
map:student_list__label a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:student_list;
    d2rq:property rdfs:label;
    d2rq:pattern "student_list #@@student_list.id@@";
map:student_list_fullname a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:student_list;
    d2rq:property vocab:student_list_fullname;
    d2rq:propertyDefinitionLabel "student_list fullname";
    d2rq:column "student_list.fullname";
map:student_list_date_of_birth a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:student_list;
    d2rq:property vocab:student_list_date_of_birth;
    d2rq:propertyDefinitionLabel "student_list date_of_birth";
    d2rq:column "student_list.date_of_birth";
    d2rq:datatype xsd:date;
map:student_list_group_number a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:student_list;
    d2rq:property vocab:student_list_group_number;
    d2rq:propertyDefinitionLabel "student_list group_number";
    d2rq:column "student_list.group_number";
map:student_list_id a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:student_list;
    d2rq:property vocab:student_list_id;
    d2rq:propertyDefinitionLabel "student_list id";
    d2rq:column "student_list.id";
    d2rq:datatype xsd:integer;
```

Запуск D2R-server. Після отримання mapping файлу з'являється можливість представити базу даних, як реляційне RDF-сховище. Отже, з'явиться можливість працювати з інформацією з реляційної бази даних як з RDF графом.

Для запуску сервера на локальному комп'ютері необхідні два вхідні параметри: mapping файл і номер порту, який буде задіяний сервером (за замовчуванням: 2020). Запущений сервер зображений на малюнку 4.3.

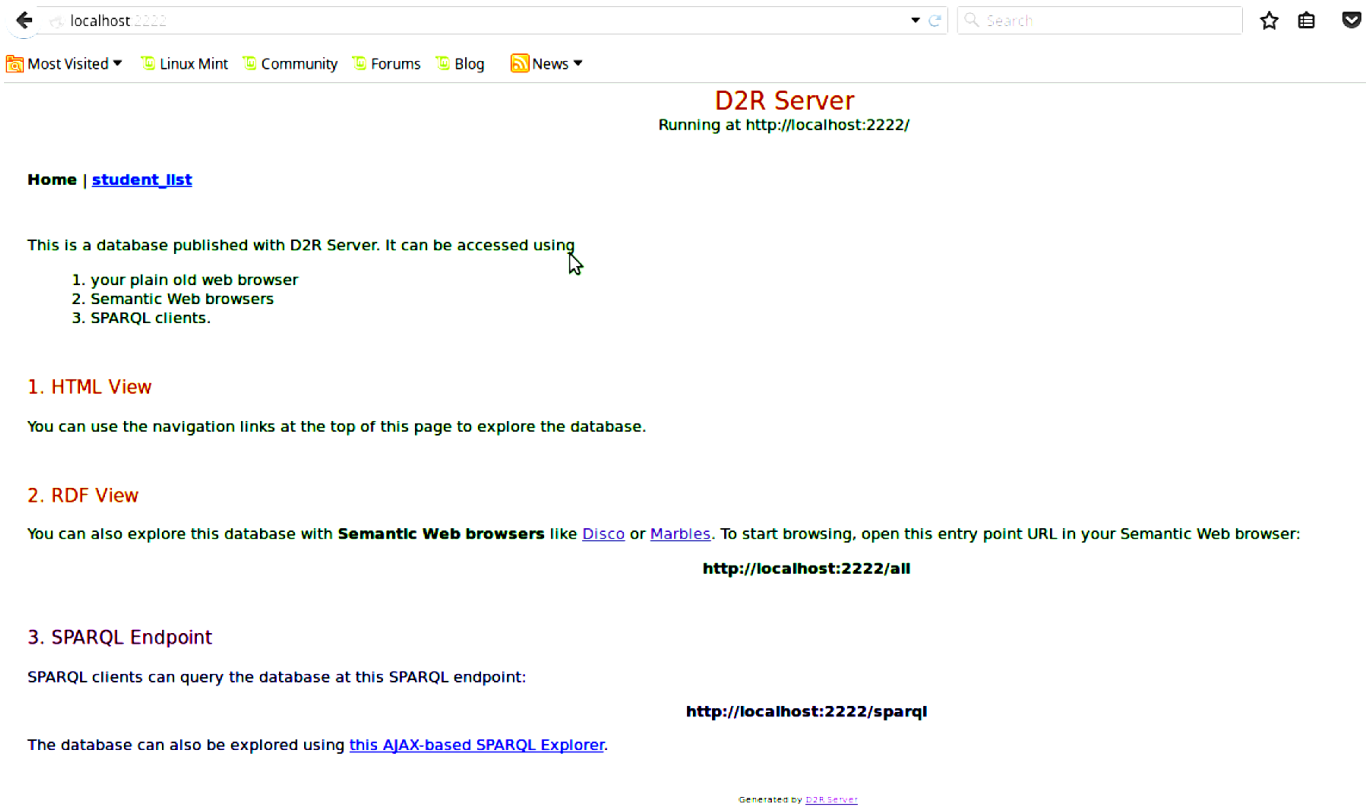

Рис. 4.3. Запуск локального D2R-сервера

Dump-rdf. Представляє дані з d2r-server у файл, де вся інформація розписана за триплетами <суб'єкт><предикат><об'єкт>. Для виклику необхідні два вхідні параметри: mapping файл і URI D2R сервера. Вихідний параметр: файл із триплетами.

```
~/java_workspace/Jena/d2rq $ ./dump-rdf -f N-TRIPLE -b http://localhost:2222/ osu_mapping.ttl > osu.nt
```

Рис. 4.4. Створення RDF дампа

Таблиця 4.2

Приклад змісту RDF дампа

```
<http://localhost:2222/student_list/1> <http://localhost:2222/vocab/student_list_id>
"1"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://localhost:2222/student_list/1> <http://localhost:2222/vocab/student_list_group_number>
     "TK-247M".
<http://localhost:2222/student_list/1> <http://localhost:2222/vocab/student_list_date_of_birth>
     "2001-06-16".
^^<http://www.w3.org/2001/XMLSchema#date> .
<http://localhost:2222/student_list/1> <http://localhost:2222/vocab/student_list_fullname>
     "Stepaniuk Andrii Vitaliovych".
<http://localhost:2222/student_list/1> <http://www.w3.org/2000/01/rdf-schema#label> "student_list #1".
```

4.3. Реалізація клієнтської частини веб-сервісу

Клієнтська частина складається з двох HTML сторінок. На початковій сторінці, зображеній на малюнку 4.5, розташовано блок інформації про веб-сервіс і його використання, а також HTML форма для надсилання URL точок доступу для реляційних баз даних.

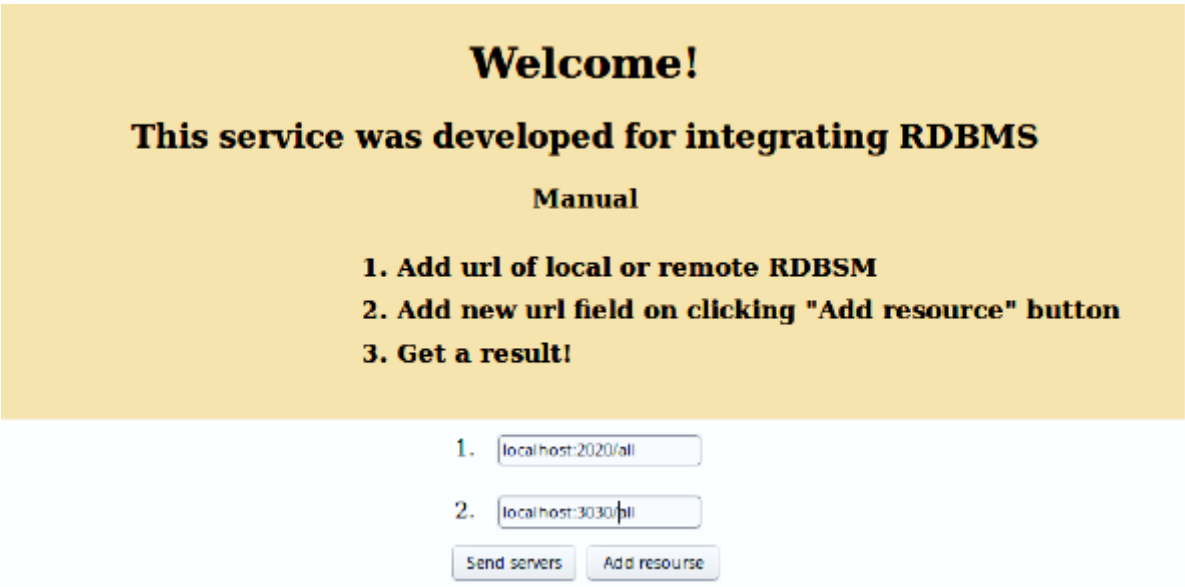

Рис. 4.5. Головна сторінка веб-сервісу

На сторінці з результатом виводиться інформація про виконану інтеграцію, а також розташовуються кнопки для скачування mapping-файлу або RDF-графа, отриманих під час інтеграції. Вони можуть бути корисними для додавання триплетів уже в наявні RDF-сховища.

Також у реалізації клієнтської частини використовується JavaScript для валідації полів і надсилання форм, у разі успішної валідації, а також додавання полів для нових точок доступу реляційних баз даних. Приклад помилки валідації зображено на малюнку 4.6.

Рис. 4.6. Приклад помилки валідації: порожня форма

РОЗДІЛ 5
ОХОРОНА ПРАЦІ

5.1. Аналіз небезпечних і шкідливих факторів, що впливають на інженера

Проєктний відділ розташований на другому поверсі п’ятиповерхового будинку. Приміщення має розміри 8 метрів на довжину, 4 метри на ширину та 4 метри на висоту. Загальна площа 32 кв. м., а загальний об’єм 128 кв. м. У відділі є п’ять робочих місць для інженерів-проєктувальників з комп’ютерами.

Робота для одного працівника становить:

$$S_{\text{роб}} = \frac{S_{\text{заг.пл}}}{N} = \frac{32}{5} = 6{,}4 \text{ м}^2$$

Обсяг роботи одного працівника:

$$V_{\text{роб}} = \frac{V_{\text{заг.об}}}{N} = \frac{128}{5} = 25{,}6 \text{ м}^3$$

N- кількість співробітників у відділі

$S_{\text{заг.пл}}$ – загальна площа;

$V_{\text{заг.об}}$ – загальний об’єм.

Згідно з [41] робоче місце має мати не менше 6 м2 і 20 м3. Робота інженера-проєктувальника відповідає стандартам.

Комп’ютери та принтери знаходяться в проєктному відділі інженера-проєктувальника. У цьому приміщенні використовується як штучне, так і природне освітлення; у теплий період року температура повітря становить тридцять градусів Цельсія. Штучне освітлення складається з переривчастих ліній світлодіодних ламп. Згідно з Державними санітарними нормами [42], рівень шуму в приміщенні становить 54 дБ.

Робоче місце розташоване таким чином, щоб природне світло падало з лівої сторони, а відстань між світлом і робочим місцем становить 1 метр. Столи мають висоту 750 мм над

підлогою, глибину 800 мм і ширину 1300 мм. Робочий стіл має місце для ніг шириною 600 мм і висотою 650 мм.

Перелік шкідливих і потенційно небезпечних виробничих факторів

Створення сприятливих робочих місць для інженерів-проєктувальників має велике значення як для полегшення, так і для підвищення продуктивності. Відповідно до [43] шкідливі виробничі фактори включають:

1. Висока температура в робочому місці.

2. Низька освітленість робочої поверхні

3. Шум виробництва.

4. Електромагнітні випромінювання діапазону радіочастот.

5. Іонізуючі речовини.

Відповідно до [44] робота інженера-проєктувальника в приміщенні з енерговитратами 90-120 ккал/год відноситься до категорії легких фізичних робіт Іа (роботи, які виконуються сидячи і не потребують фізичного напруження).

Таблиця 5.1

Оптимальні величини температури

Період року	Категорія робіт	Температура повітря,°С
Холодний період року	Легка Іа	22-24
Теплий період року		23-25

Допустимі величини температури на постійних робочих місцях:

Період року	Категорія робіт	Температура повітря,°С	
		Верхня межа	Нижня межа
Холодний період року	Легка Іа	25	21
Теплий період року		28	22

Допустимі величини температури на постійних робочих місцях:

Період року	Категорія робіт	Температура повітря,°С	
		Верхня межа	Нижня межа
Холодний період року	Легка Іа	25	21
Теплий період року		28	22

У теплий період року температура повітря в проєктному відділі перевищує допустиму на 2 °С. Вентилятор VORTICE VARIO забезпечує механічну вентиляцію з повітрообміном 680 м^3/год і температурою приміщення 23 °С.

Недостатня освіта В приміщенні є штучне та природне освітлення, встановлені комп'ютери. Як вимагається [45], коефіцієнт природної освітленості не повинен перевищувати 1,5 відсотка. Освіченість робочої поверхні в проєктному відділі порушення вимоги складає 370 лк, а коефіцієнт освітленості складає 1.2%. Бічні прорізи дозволяють природному світлу проникати в приміщення. У вікнах є жалюзі. Штучне освітлення складається з переривчастих ліній світлодіодних світильників, розташованих паралельно лінії зору інженера-проєктувальника. Використовуйте галогенні лампи розжарювання для місцевого освітлення. Комп'ютери та периферійні пристрої створюють шум на робочому місці. Рівень звукового тиску, дозволеного на робочому місці, повинен відповідати стандартам [46]:

Таблиця 5.2

Санітарні норми виробничого шуму, ультразвуку та інфразвуку

Вид трудової діяльності, робоче місце	Рівні шуму та еквівалентні рівні шуму, ДБА, дБАекв
Конструювання та проектування.	50

Рівень шуму в проєктному відділі перевищує норму на 54 дБ.

Якщо ви хочете зменшити рівень шуму, ви повинні використовувати як місцеву, так і загальну звукоізоляцію, шумопоглинаючі екрани та поглинаючі фільтри.

5.2. Організаційно-технологічні та конструктивні заходи для зменшення впливу шкідливих виробничих факторів

Нормалізація повітря на робочому місці. Для створення та автоматичної підтримки в ІТ-відділі необхідно дотримуватися ідеальних температур, вологості, чистоти та швидкості руху повітря незалежно від зовнішніх умов. У холодні роки використовується водяне опалення, а в теплі роки використовується кондиціонування повітря [47].

Технічне освітлення Під час аналізу освітлення робочого міста програміста було виявлено, що воно не відповідає встановленим нормам. Тому, щоб покращити умови праці, ми рекомендуємо встановити п’ять додаткових світильників і збільшити загальну кількість лам до 36 світлодіодних ламп. Крім того, необхідно планувати очищення віконних блоків і світильників не менше двох разів на рік, щоб забезпечити чисте освітлення [48].

Безпека електроенергії. Пропоную наступні технічні заходи та засоби захисту для забезпечення електробезпеки в ІТ-відділі: зволожувачі та нейтралізатори, антистатичне покриття підлоги; зменшення накопичення статичної електрики; забезпечити приєднання металевих корпусів устаткування до жили, що заземлюється. Заземлення корпусу комп’ютера дозволяє підвести жилу, що заземлює, до розеток. Для електроустановок з напругою до 1000 В стандартний опір заземлення становить 4 Ом. Крім того, заходи з організації, такі як забезпечення своєчасного проведення інструктажів з техніки безпеки [49].

Ергономіка та управління робочим місцем Після оцінки робочого місця програміста в ІТ-відділі було встановлено, що воно відповідає вимогам.

Виходячи з результатів аналізу важкості та напруженості праці, пропоную скоротити час роботи за комп’ютером до п’ятдесяти хвилин на робочому дні з восьми годинним робочим днем [50].

5.2.1. Розрахунок повітрообміну за надлишком тепла у проєктному відділі

Приміщення розміром 4 на 8 на 4 метрів знаходиться на другому поверсі п’ятиповерхового будинку, розташованого з південного боку будинку. Ф = 2,88 м^2. Жалюзі на вікнах. У приміщенні знаходяться п’ять інженерів-проєктувальників $N_{пк}$=5 комп’ютери та принтери. Щоб забезпечити штучне освітлення, використовується чотири офісних світлодіодних світильника потужністю 125 Вт.

1. Розраховуємо загальну кількість тепла:

$$Q_{над} = Q_{осв} + Q_{облад} + Q_{ін\text{-}пр.} + Q_{рад,} \text{ Вт} \quad (5.1)$$

$Q_{над}$ – загальна кількість тепла

$Q_{осв}$ - кількість тепла від джерел штучного освітлення

$Q_{облад}$ - кількість тепла від обладнання

$Q_{ін\text{-}пр.}$- кількість тепла від інженерів-проєктувальників

$Q_{рад,}$- кількість тепла від сонячної радіації

2. Розраховуємо кількість тепла від джерел штучного освітлення:

$$Q_{осв} = N \cdot \eta, \quad (5.2)$$

де N - сумарна потужність джерел освітлення, Вт; η - коефіцієнт теплових витрат (η = 0,55 – для світлодіодних ламп).

$$Q_{осв.} = 125 \cdot 4 \cdot 0,55 = 275 \text{ Вт}$$

2. Розраховуємо кількість тепла при роботі обладнання: 5 комп’ютерів і принтера (в режимі друку):

$$Q_{облад} = n \cdot P_{комп.} + P_{пр.}, \quad (5.3)$$

де n – кількість комп’ютерів (обладнання);

$P_{комп}$ – встановлена потужність комп'ютерів, $P_{комп}=400$ Вт

$P_{пр.}$ – потужність принтера в режимі друку, $P_{пр}=465$ Вт

$$Q_{облад}=5\cdot 400+465=2.5 \text{ кВт}$$

3. Розраховуємо кількість тепла від інженерів-проєктувальників:

$$Q_{ін\text{-}пр.} = n \cdot q \text{ , Вт} \tag{5.4}$$

n – кількість інженерів-проєктувальників

q – кількість тепла, що виділяється одним інженером-проєктувальником

Кількість тепла, що виділяється одним інженером-проєктувальником, який виконує легку фізичну роботу дорівнює 99 Вт.

$$Q_{ін\text{-}пр} = 5 \cdot 99 = 495 \text{ Вт}$$

4. Розраховуємо кількість тепла від сонячної радіації:

$$Q_{рад}=m \cdot S \cdot k \cdot q_{скл} \tag{5.5}$$

де m – число вікон; $S_{вікна}$ – площа одного вікна, $S_{вікна}$ =2,88 $м^2$;

k – коефіцієнт, віконного переплетення: k = 0,6 матові;

$q_{скл.}$ – надходження тепла через 1 $м^2$ вікна при різній орієнтації вікон: $q_{скл.}$ = *150* – південь;

$$Q_{рад}=1 \cdot 2,88 \cdot 0,6 \cdot 150=259,2 \text{ Вт}$$

5. Загальна кількість тепла в проєктному відділі:

$$Q_{над}= Q_{осв}+ Q_{облад}+ Q_{ін\text{-}пр.} + Q_{рад}=275+2500+495+259,2 = 3,529 \text{ кВт}$$

6. Потрібний повітрообмін за надлишком тепла:

$$L = \frac{Q}{c \cdot \rho \cdot (t_{\text{вид}} - t_{\text{зовн}})}, \text{м}^3/\text{год} \quad (5.6)$$

Q - кількість тепла, яке виділяється в приміщення за годину, Дж:

$$Q = 3600 \cdot Q_{надл} = 3600 \cdot 3529 = 12704 \text{ Вт} = 5328 \text{ кДж};$$

c – теплоємність повітря, Дж/кг (в інтервалі температур від 0ºС до 100ºС приймається рівною $1{,}01 \cdot 10^3$Дж/кг);

ρ – густина повітря, кг/м^3(дорівнює $\rho_{внт}$=1,2 кг/м^3);

$t_{вид}$– температура повітря, що видаляється, $t_{вид}$=30℃

$t_{зовн.}$- температура повітря, що подається до робочої зони, $t_{зовн.}$=23℃

$$L = \frac{5328}{1{,}01 \cdot 10^3 \cdot 1{,}2 \cdot (30 - 23)} = 628 \text{ м}^3/\text{год}$$

У проєктному відділі температура повітря підвищена на 2 °С від допустимої 28 °С, тому було встановлено механічну вентиляцію з вентилятором VORTICE VARIO, який забезпечує температуру повітря в приміщенні 23 °С, що є оптимальною температурою.

5.3. Пожежна безпека

Відповідно до [51], через використання твердих горючих матеріалів з температурою спалаху понад 61°С це приміщення відноситься до категорії В по вибухово-пожежній та пожежній небезпеці.

Проєктний відділ оснащений двома безпровідними датчиками диму SD-02, які повідомляють про задимлення в приміщенні. Вони мають площу обслуговування до 20 м2. Крім того, є два порошкові вогнегасники ВП-5, які призначені для приміщень

категорії В, які не містять горючих газів і рідин, які мають площу до 50 м2 і масу вогнегасної речовини 5 кг, з мінімальною кільк

Бездротова система пожежно-охорони LifeSOS LS-30LR використовує радіоканал, щоб передавати сигнал тривоги на центральний блок при виявлення вторгнення. Використовуючи сигнали датчиків, включаючи сирену, центральний оператор передає інформацію на пульт централізованого нагляду, дзвонить на зазначені телефонні номери та відправляє повідомлення про тривогу через SMS.

Організаційно-технічні заходи щодо пожежної безпеки включають включення питань пожежної безпеки у всі інструкції по техніці безпеки; дотримання правил експлуатації електричних мереж і обладнання; заборона куріння в недозволених місцях; і надання необхідних інструктажів і планів евакуації.

План евакуації складається з текстових і графічних елементів. Згідно з рис. 5.1, схематичний план поверху представлений зеленими суцільними стрілками, які показують шляхи евакуації, які ведуть до основних евакуаційних виходів, а також аварійними виходами. Двері евакуаційного шляху відчиняються у напрямку виходу з будівлі. Умовні знаки на плані евакуації показують, де знаходяться вогнегасники, пожежні гідранти, телефони, аптечки медичної допомоги, електрощити, датчики диму та системи охоронно-пожежної сигналізації.

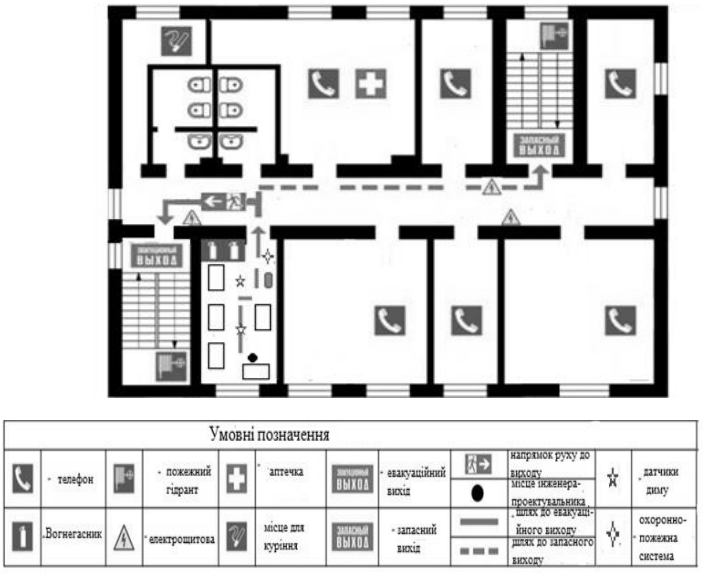

Рис 5.1. План евакуації 2 поверх

5.4. Інструкція з охорони праці при роботі з персональним комп'ютером

Попередні заходи безпеки перед початком роботи:

- Перед початком роботи працівник повинен перевірити цілісність корпусу системного блоку, принтера, клавіатури та відео монітора.
- Перевірте цілісність кабелів живлення та місць їх підключення, включаючи розетки, продовжувачі, розгалужувачі та штепсельні вилки.
- Підготувати робоче місце, прибравши все, що заважає вам виконувати завдання.
- Включити живлення ПК.
- Працівник повинен повідомити керівника або спеціаліста відділу інформаційних технологій, якщо комп'ютер не завантажується або не виходить на робочий режим після ввімкнення.
- Повідомте безпосереднього керівника про будь-які проблеми. не починати роботу без його дозволу.

Вимоги до безпеки під час роботи:

- Усі компоненти столу, включаючи клавіатуру, повинні бути стійко розташовані. Крім того, необхідно передбачити можливість переміщення клавіатури. Її розміщення та кут нахилу повинні відповідати потребам користувача РК. Якщо в конструкції клавіатури немає місця для опору долонь, клавіатура повинна бути розташована на відстані не менше 100 мм від краю столу, щоб забезпечити оптимальну відстань для моніторного поля. Коли ви працюєте на клавіатурі, вам слід сидіти прямо, не напружуючись.
- Для зменшення несприятливого впливу на користувача пристроїв типу «миша» (вимушена поза, постійний контроль за якістю дій) слід забезпечити більшу площу поверхні столу, щоб «миша» могла переміщатися і мати зручний упор для ліктьового суглоба.
- Не дозволяється розмовляти сторонніми речами, створювати шуми тощо.

- Кожного разу, коли комп’ютер не працює, хлопко-паперова салфетка слід мити з мильним розчином. Екран і захисний екран протирають спиртом.
- Не дозволяється використовувати рідинні або аерозольні засоби для очищення поверхонь комп’ютерних технік.

Забороняється:

- самостійно ремонтувати апаратуру, в якій кінескоп та інші частини можуть бути під високою напругою (до 25 кВ0);

- класти будь-які предмети на клавіатуру або поруч з нею, включаючи бутерброди та напої; Це може вивести її з ладу;

- затуляти вентиляційні отвори в пристрої, що може призвести до перегріву та виходу з ладу.

Щоб зменшити негативний вплив факторів ризику, пов’язаних із роботою на комп’ютерах, на стан здоров’я працівників, передбачаються додаткові регламентовані перерви для відпочинку користувачів комп’ютерів: 10 хвилин під час безперервної роботи та 15 хвилин під час кожної другої години роботи.

Якщо це можливо, слід змінити діяльність на іншу, яка не пов’язана з роботою на ПК.

Чергування операцій введення тексту та введення даних (зміна змісту та темпу роботи) може зменшити негативний вплив монотонності.

При роботі з лазерными принтерами слід переконатися, що принтер розташований поруч із системним блоком, щоб з’єднувальні шнури не натягнулися. Не можна встановлювати принтер на системний блок.

Перш ніж розпочати програмування принтера, переконайтеся, що він знаходиться в режимі зв’язку з системним блоком.

Щоб уникнути пошкодження апарату, потрібно використовувати папір, марка якого вказана в інструкції до принтера. Це найчастіше папір вагою 60-135 г/м2, типу Canon або Xerox 4024.

Щоб зменшити ймовірність загинання паперу, обрізайте краї паперу гострим лезом ножа, який не має заусенців.

Бажано вимикати живлення відео монітора під час роботи (більше 20 хвилин), коли втручання користувача в програму не потрібне.

Під час перерв необхідно виконувати рекомендовані вправи для очей, хребта та рук, щоб підтримувати загальний тонус м'язів і запобігти кістково-м'язовим проблемам, зоровому дискомфорту та іншим несприятливим суб'єктивним почуттям.

Кількість мікропауз (від 1 до 2 хвилин) має бути індивідуальною. Перерви можуть мати різний характер і включати виконання допоміжних робіт, не пов'язаних із роботою ПК, прийом їжі та виконання рекомендованих вправ.

Залежно від того, наскільки ви втомилися, рекомендується виконувати фізичні вправи протягом дня. Гімнастика повинна сприяти корекції вимушеної пози, покращити кровообіг, частково компенсувати та скоротити рухову активність.

Якщо ви помітите будь-яку несправність, таку як іскри, пробоїв, запах гару або ознаки горіння, негайно припиніть роботу, відключіть все обладнання від електромережі та негайно повідомте безпосереднього керівника або спеціаліста по ремонту комп'ютерів.

Правила безпеки для завершення роботи на персональному комп'ютері:

Закінчити та зберегти файли, які знаходяться в роботі в пам'яті комп'ютера (ПК). Виконати всі необхідні дії, щоб забезпечити коректне завершення роботи операційної системи.

Вимкнути системний блок і принтер, а також інші периферійні пристрої. Вимкнути живлення пристрою безперебійного живлення (ПБЖ).

Вимкнути комп'ютер кнопкою «POWER» (ЖИВЛЕННЯ) і вийняти штепсельну вилку з кабелю живлення.Накрийте клавіатуру кришкою, щоб уникнути попадання пилу.

Порядок на робочому місці:

Стандарти безпеки в аварійних ситуаціях:

Негайно відключити комп'ютер від електромережі та повідомити про це своєму керівникові, якщо він виділяє запах горілого або має металеві частини.

Якщо виникне пожежа, негайно розпочати гасіння наявними засобами пожежогасіння та повідомити за номером 101 (міська пожежна охорона) і начальнику відділу безпеки підприємства. Пам'ятайте, щоб уникнути ураження електричним струмом, загасіть електроустановки вуглекислотними вогнегасниками або сухим піском.

У разі інших аварійних подій роботу слід припинити та повідомити про це керівника робіт.

Висновки. Розрахунки показали, що повітрообмін за надлишком тепла становить 628 м3/год, що вказує на те, що використання природної вентиляції є малоефективним. З цієї причини ми вибрали механічну вентиляцію з вентилятором VORTICE VARIO. Механічна вентиляція може забезпечити вихід з проєктного відділу температури 30 градусів Цельсія та підтримувати температуру повітря в межах дозволеного та навіть ідеального рівня.

РОЗДІЛ 6
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1. Аналіз впливу техногенних чинників

Широке використання електричного та електронного обладнання призвело до негативних наслідків для здоров'я людини та навколишнього середовища. Є деякі основні шкідливі та небезпечні фактори, які впливають на навколишнє середовище [52]: шумове, вібраційне, електромагнітне, теплове та радіаційне забруднення.

Злочинне забруднення Сьогодні завдяки науково-технічним досягненням шум став однією з форм фізичного (хвильового) забруднення природного середовища. Усі неприємні та небажані звуки або їхня сукупність вважаються шумом, оскільки вони заважають людям працювати, сприймати потрібну звукову інформацію та відпочивати.

До нього практично неможливо адаптуватися. Рівень фонового шуму навколишнього середовища становить від тридцяти до сорока децибел. Сьогоднішні виробничі та транспортні шуми нерідко перевищують 100 децибел, доповнюючи природний фон. Шум може походити від промислових об'єктів, транспорту, гучномовних пристроїв, музичних інструментів, телевізорів, радіоприймачів і юрб людей. У виробничих умовах шум негативно впливає на працівників, оскільки він послаблює їхню увагу, сприяє втомі та сповільнює їхню реакцію на небезпеку. Як наслідок, працездатність знижується, а кількість нещасних випадків збільшується. Таблиця 5.1 містить допустимі рівні звукового тиску в октавних смугах частот для робочих місць у виробничих приміщеннях [52]:

Таблиця 6.1.

Допустимі рівні звукового тиску в октавних смугах частот

Рівні звукового тиску в дБ, в октавних смугах частот, Гц								
31,5	63	125	250	500	1000	2000	4000	8000
107	95	87	82	78	75	73	71	69

Вплив шуму знижує енергію до зростання рослин і викликає надмірне виділення вологи через листя, що може призвести до смерті. Листя та квіти рослин, які знаходяться в непосредственній близькості від джерела інтенсивного шуму, гинуть. Відсутність шуму особливо важлива для тварин, які обмінюються звуковою інформацією, а також для тварин, які аналізують звуки навколишнього середовища, щоб отримати більше інформації, включаючи сигнали тривоги. Тварини також відчувають шум. Личинки бджіл втрачають здатність орієнтуватися, а шкаралупа яєць тріщить у пташиних гніздах. Жуки, джмелі та інші комахи не можуть вдихати коливання повітря, створені переносною радіоапаратурою.

Забруднення від вібрації Вібрації є механічними коливаннями твердих тіл. Вібрація природна та штучна. Землетруси, викликані природними факторами, є джерелом природної вібрації. Штучна вібрація походить від бізнесу та транспорту. Тривалі вібрації шкодять здоров'ю людини, викликаючи сильну втому та змінюючи багато функцій організму. Це включає порушення серцевої діяльності, нервової системи, спазм судин, деформацію м'язів, струси головного мозку тощо. Вібрація високої частоти може пошкодити певні органи чи частини тіла людини. Вібраційна хвороба — це професійне захворювання, яке може виникнути через постійну вібрацію [52].

Електромагнітна шкода Електромагнітне поле (ЕМП) природного походження, також відоме як природний фон, завжди впливало на біосферу протягом усього процесу еволюції. Ці джерела включають електромагнітне поле Землі та космічне електромагнітне випромінювання, особливо те, що виробляється Сонцем. У міру розвитку технологій людина створювала та використовувала штучні (антропогенні) джерела

ЕМП. У наш час антропогенні ЕМП значно перевищують природний фон і є тим негативним фактором, вплив якого на людину та довкілля зростає щороку. Діапазон частот, інтенсивність і тривалість дії, тип випромінювання (неперервний чи модульований), режим опромінювання, розмір поверхні тіла, що зазнає опромінювання, і особливості організму впливають на вплив ЕМП на організм людини. Електромагнітні поля можуть порушити біологічні та функціональні процеси організму, що впливає на його функціонування. Передчасна втомлюваність, часті болі голови, поганий сон і порушення роботи серцево-судинної та центральної нервової системи є ознаками функцій. ЕМП викликають хронічні захворювання та порушення. Вплив ЕМП має біологічні негативні наслідки як у тепловій, так і в нетепловій діях. Під час теплової дії електромагнітна енергія перетворюється на теплову, що призводить до підвищення температури тіла та локального нагрівання тканин і органів. Таке нагрівання особливо небезпечно для органів, які мають погану терморегуляцію, таких як шлунок, нирки, очі та головний мозок. Наприклад, катаракта, або поступова втрата зору, може виникнути в результаті випромінювання сантиметрового діапазону [52].

Забруднення теплом Термін «теплове забруднення» стосується теплоти, яка виділяється в навколишнє середовище під час низки теплових процесів, головним чином згорання палива. За рік згорання палива витрачається до 23 відсотків усього кисню, що утворюється в процесі фотосинтезу на Землі. У разі безаварійної роботи атомних електростанцій під час спалювання вугілля в навколишнє середовище викидається більше радіоактивних речовин. ТЕС, АЕС та інші енергетичні об’єкти скидають підігріті води у водойми, що призводить до теплового забруднення гідросфери. Тепла вода шкідливо впливає на мешканців водойм, змінюючи температурні та біологічні режими [52].

6.2. Вплив приймальних пристроїв на навколишнє середовище

Абонентський приймач — це пристрій, який приймає цифровий телевізійний сигнал, декодує його, перетворює його на аналоговий сигнал через роз’єми RCA або

SCART або перетворює його на цифровий сигнал через роз'єми HDMI і передає його далі на телевізор.

Перехід до цифрового телебачення призвів до зростання виробництва цифрових абонентських приймачів, що може мати шкідливі наслідки для навколишнього середовища. Приймач генерує слабкі електричні та магнітні змінні поля, які працюють у широкому діапазоні частот. Однак проблема впливу електромагнітних випромінювань, що продукуються, повинна бути вирішена. Наукові дослідження показали, що ЕМВ мають вплив на користувачів сучасних екранів із ЕМВ. Вчені з України визначили це явище як торсіонові поля, які супроводжують електромагнітне випромінювання та є його інформаційною частиною [53]. Робоча група Всесвітньої організації охорони здоров'я, яка вивчала гігієнічні аспекти користування моніторами та радіо терміналами, виявила, що користування такими пристроями спричиняє проблеми зі здоров'ям, найсерйозніші з яких включають погіршення зору, проблеми з імунною системою та проблеми з психоемоційною сферою (стрес, агресія).

Державні санітарні норми і правила при роботі з джерелами електромагнітних полів (ДСанПіН 3.3.6.096-2002) діють в Україні, щоб гарантувати безпеку користувачів. Таблиця 5.2 містить значення ГДР напруженості електричної (Егд) і магнітної (Нгд) компонентів залежно від тривалості їх дії.

Таблиця 6.2.

Значення ГДР напруженості електричної ($E_{гд}$) і магнітної ($H_{гд}$) складових

Час перебування персоналу, год	$E_{гд}$, В/м					$H_{гд}$, А/м			
	1-10 кГц	10-60 кГц	0,063 МГц	3-30 МГц	30-300 МГц	1-10 кГц	10-60 кГц	0,06-3 МГц	30-50 МГц
8	120	70	50	30	10	9	7	5	0,3
7	130	75	53	32	11	9,8	7,5	5,3	0,32
6	140	82	58	34	12	10,6	8,1	5,8	0,34
5	155	90	63	37	13	11,6	8,8	6,3	0,38
4	175	110	71	42	14	13	9,9	7,1	0,42
3	200	115	82	48	16	15	11,4	8,2	0,49
2	250	140	100	59	20	18,4	14	10	0,6
1	350	200	141	84	28	26	19,7	14,2	0,85
0,5	500	280	200	118	40	37,6	27,9	20	1,2

Електромагнітні випромінювання в діапазоні 30 кГц–300 МГц (НЧ) можуть спричинити загальну слабкість, підвищену втому, сонливість, порушення сну, головний біль і біль у серці. Роздратованість, невпевненість і сповільнені рухово-мовні реакції Ряд симптомів вказує на порушення роботи певних органів, таких як шлунок, печінка та підшлункова залоза.

Обмеження кількості електромагнітного випромінювання, яке виділяється абонентським приймачем, необхідно для зниження рівня електромагнітного випромінювання [54-55].

Державні санітарні норми і правила захисту населення від впливу електромагнітного випромінювання визначають стандарти електромагнітної безпеки в Україні. Ці стандарти встановлюють допустимі рівні інтенсивності електромагнітного випромінювання для населення в межах 2,5 мкВт/см2.

Під час роботи абонента приймач генерує шум 54 дБ. Рівень звукового тиску, дозволеного згідно з «ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку», має становити 50 дБ.

Багато звукових сигналів потрапляють до кори головного мозку, що викликає тривогу, тривогу та передчасну втому. Коли шум впливає на людину, він може спричинити зміни в ЦНС, органах слуху, серцево-судинній та ендокринній системах, травленні та інших органах і системах, а також суб'єктивні порушення. Скарги на роздратованість, переживання та порушення сну є першими ознаками шкідливої дії шуму [56].

6.3. Засоби для захисту від електромагнітного випромінювання та шуму, проблема електронних відходів

Захист від електромагнітних пошкоджень. Потрібно вжити низку заходів безпеки, щоб зменшити вплив ЕМП на працівників і інших людей, які проживають у зоні дії радіоелектронних засобів. Це можуть бути організаційні, інженерно-технічні та лікарсько-профілактичні.

Організаційні, інженерно-технічні та лікарсько-профілактичні заходи зменшують вплив на працівників ЕМП.

Органи санітарного нагляду відповідають за організацію. Вони стежать за станом здоров'я людей на місцях, де використовуються джерела електромагнітних випромінювань.

Інженерно-технічні заходи включають розташування джерел ЕМП таким чином, щоб їх вплив на працівників був мінімальним; використання дистанційного керування апаратурою, що є джерелом випромінювання, екранування джерел випромінювання; застосування засобів індивідуального захисту, таких як халати та комбінезони із металізованої тканини з виводом на заземлюючий пристрій; і використання апаратури, що є джерелом випромінювання Додатковим засобом захисту очей є захисні окуляри ЗП5-90. Напівпровідникове олово, яке вкрито склом окулярів, зменшує інтенсивність електромагнітної енергії при світлопропусканні не нижче 75%.

Взагалі кажучи, засоби індивідуального захисту слід використовувати лише тоді, коли інші засоби захисту неможливі або недостатньо ефективні, наприклад, під час проходження через зони опромінення підвищеної інтенсивності, під час ремонту та налагодження в аварійних ситуаціях, під час короткочасного контролю та коли інтенсивність опромінення змінюється. Такі інструменти незручні в експлуатації, обмежують можливості виконання операцій на робочому місці та погіршують гігієнічні умови.

У радіочастотному діапазоні засоби індивідуального захисту захищають людину за допомогою відбиття та поглинання ЕМП. Для захисту тіла використовується одяг, виготовлений з металізованих тканин, а також матеріали, які запобігають проникненню рідини. Металізована тканина може бути виготовлена з бавовняних ниток, у яких є тонкий провод, або з капронових чи бавовняних ниток, які спірально обвитий металевим дротом. Така тканина, схожа на металеву сітку, значно послаблює дію випромінювання при відстані між нитками до 0,5 мм. Забезпечте, щоб ізольовані провода не контактували під час зшивання компонентів захисного одягу. Таким чином, електрогерметизація швів проводиться за допомогою електропровідних мас або клеїв,

які забезпечують гальванічний контакт або збільшують ємність безконтактних проводів.

Лікарсько-профілактичні заходи включають регулярні медичні огляди працівників, які перебувають у зоні дії ЕМП; обмеження тривалості перебування людей у зоні підвищеної інтенсивності електромагнітних випромінювань; надання працівникам безкоштовного лікарсько-профілактичного харчування; і перерви, пов'язані з санітарними потребами.

Щоб захистити від шуму Шумозахист — це комплекс заходів, призначених для зменшення та ліквідації шуму. Це включає використання звукопоглинаючих матеріалів, правильне розміщення будівельних об'єктів, створення земляних валів вздовж вулиць, стін різних конструкцій і застосування шумопоглинаючих матеріалів, які здебільшого не стосуються житлових будівель, таких як магазини, склади та гаражі.

Проблема з відходами Згідно з Законом про відходи, необхідно створити системи збирання та утилізації електричного та електронного обладнання з метою зменшення або запобігання утворенню відходів [55-56]. «Технічний регламент з поводження з відходами електричного та електронного обладнання», розроблений в Україні з 2008 року, має вирішити проблему електронних відходів. Згідно з проєктами цих законів виробники та імпортери можуть як самостійно утилізувати електровідходи, так і укласти договори з уповноваженими підприємствами щодо збирання, заготівлі та утилізації відповідних видів техніки. Крім того, розроблено проєкт Постанови Кабінету Міністрів України «Про затвердження Технічного регламенту з поводження з відходами електронного та електричного устаткування». Регламент передбачає створення місць збору електронного та електричного обладнання. Ці місця повинні розташовуватися у зручних місцях для користувачів і надавати безоплатні послуги. Наразі розглядається ще один спосіб вирішення проблеми. Це внесення змін до Податкового кодексу, який дозволить стягувати гроші з імпортерів і виробників різних споживчих товарів з метою забезпечення належного управління збиранням, заготівлею та утилізацією відходів від цих товарів.

Тим не менш, загалом проблема електронних відходів в Україні повинна бути вирішена з організаційно-правової точки зору, наприклад, створивши фонди для виробників, фінансуючи державні підприємства з утилізації відходів, а також використовуючи соціальні мережі, щоб переконати громадян, що викидати поламані електронні пристрої на смітник неправильно.

Висновок. Інженерно-технічні заходи, які зменшують дію шкідливих факторів, є корисними, щоб зменшити ризик виникнення захворювань. Крім того, було розглянуто питання електронних відходів; одним із способів вирішення проблеми є створення місць збору електронного та електричного обладнання.

ВИСНОВКИ

Тема "Метод інтеграції реляційних баз даних із використанням платформи D2RQ" є актуальною в контексті сучасних тенденцій в області інформаційних технологій, де спостерігається зростання інтересу до семантичного вебу та інтеграції різних джерел даних.

Платформа D2RQ виявилася ефективним інструментом для мапінгу реляційних баз даних до формату RDF, що дозволяє інтегрувати існуючі реляційні бази даних з семантичним вебом без необхідності їх повного переформатування чи перепису. Це забезпечує можливість використання великих обсягів існуючих даних в нових, глобальних семантичних контекстах, що може стати ключовим фактором для розвитку семантичних технологій.

Водночас, використання D2RQ має деякі обмеження та особливості, які слід враховувати при плануванні проєктів інтеграції. Специфіка реляційних баз даних, їх структурні особливості та потреби користувачів можуть вимагати додаткової настройки та оптимізації процесу мапінгу.

Загалом, можна констатувати, що метод інтеграції реляційних баз даних із використанням платформи D2RQ представляє великий потенціал для розвитку семантичного вебу і розширення можливостей використання реляційних баз даних. На основі проведеного дослідження рекомендується розглядати D2RQ як один із ключових інструментів для інтеграції реляційних баз даних в сучасних IT-проєктах, пов'язаних із семантичними технологіями.

У уваліфікаційній роботі було проведено дослідження проблеми інтеграції різнорідних реляційних баз даних організації з використанням технологій роботи з семантикою. Розроблено архітектуру програмної системи, що включає використання посередників-адаптерів для перетворення реляційних БД на RDF-граф і базується на MVC-моделі проєктування програмного забезпечення. На основі використання RDF графів, було розроблено алгоритм інтеграції. Проведено дослідження можливостей

D2R, RDFLib. Створено прототип програмної системи, що включає клієнтську та серверну частини веб сервісу, а також платформи D2R для реалізації посередників-адаптерів. Проведено апробацію системи на прикладі реляційних СУБД: PostgreSQL і MySQL.

У результаті проведеного дослідження показано можливість створення ефективних систем інтеграції реляційних СУБД на основі стандартних інструментів Semantic Web.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. O. Ochoa, M. Rodney and N. Del Rio, "Accessing Provenance Records in Semantic Web Services," 2020 IEEE 14th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, 2020, pp. 141-144.
2. N. Kottekat and K. Gary, "GATE II: Visualizing Semantic Web Search," 2022 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2022, pp. 1866-1871.
3. A. Kargar and S. Emadi, "Fault Tolerance in Automatic Semantic Web Service Composition based on QoS-awareness Using BTSC-DFS Algorithm," 2019 5th International Conference on Web Research (ICWR), Tehran, Iran, 2019, pp. 50-54.
4. N. N. Akram and V. Ilango, "Intelligent Web Mining Techniques using Semantic Web," 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichy, India, 2022, pp. 1-7.
5. A. Takhom, D. Leenoi, P. Soomjinda, S. Usanavasin, P. Boonkwan and T. Supnithi, "A Supportive Environment for Knowledge Construction based on Semantic Web Technology: A Case Study in a Cultural Domain," 2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Chiang Mai, Thailand, 2019, pp. 1-4.
6. X. Chen, T. Wu, Q. Xie and J. He, "Data Flow-Oriented Multi-Path Semantic Web Service Composition Using Extended SPARQL," 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 2017, pp. 882-885.
7. E. John and M. Siddique, "Efficient Semantic Web Services Development Approaches using REST and JSON," 2021 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 2021, pp. 231-235.
8. C. S. S. Kumar, M. Mohanapriya and C. Kalaiarasan, "A new approach for information retrieval in semantic web mining involving weighted relationship,"

2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-4.

9. Z. Fayçal and T. Abdelkamel, "Building a semantic web services ontology in the pharmaceutical field using the OWL-S Language," 2021 International Conference on Information Systems and Advanced Technologies (ICISAT), Tebessa, Algeria, 2021, pp. 1-8.
10. C. Ramesh, K. V. C. Rao and A. Govardhan, "Ontology based web usage mining model," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2017, pp. 356-362.
11. S. Satveer and S. Singh, "Towards a Life Cycle Model for the Development of Semantic Web Applications," 2022 International Conference on Fourth Industrial Revolution Based Technology and Practices (ICFIRTP), Uttarakhand, India, 2022, pp. 154-159.
12. N. Ramzy, P. Ulrich, L. Mairesse and H. Ehm, "Demand Predictability Evaluation for Supply Chain Processes Using Semantic Web Technologies Use Case," 2022 Winter Simulation Conference (WSC), Singapore, 2022, pp. 1-12.
13. S. S. Zareen, S. Guangmin, S. U. Qadri and A. Qadir Khan, "Towards the Development of Framework for Semantic Web Interface," 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 2019, pp. 2723-2727.
14. M. Lian, D. Yang, Z. Yin, J. Liu, M. Li and A. Chai, "A Semantic Web Service Oriented Middleware Framework for Internet of Things," 2020 13th International Conference on Intelligent Computation Technology and Automation (ICICTA), Xi'an, China, 2020, pp. 553-557.
15. S. Kakad and S. Dhage, "Semantic Web Rule Based Decision Support System: Knowledge Graph," 2022 2nd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2022, pp. 1-6.

16. M. Gulzar and M. Ahmed, "Chronic Disease Management using Semantic Web Technologies," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023, pp. 1629-1633.
17. R. Bonacin, M. Fugini, R. Martoglia, O. Nabuco and F. Saïs, "Web2Touch 2020–21 : Semantic Technologies for Smart Information Sharing and Web Collaboration," 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 2020, pp. 235-238.
18. R. Sethuraman, G. Sneha and D. S. Bhargavi, "A semantic web services for medical analysis in health care domain," 2017 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2017, pp. 1-5.
19. R. Hammami, H. Bellaaj and A. H. Kacem, "rMatcher: A Tool for Semantic Web Services Discovery & Publication," 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Poznan, Poland, 2017, pp. 311-313.
20. G. Tian, G. Li, L. Su and X. Chen, "Services Rating Based On Privacy-Ontology in Semantic Web of Things," 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 2017, pp. 16-20.
21. J. Thiombiano, Y. Traoré, S. Malo, P. Koassa and O. Sié, "Semantic annotation of resources based on ontologies:application to a knowledge sharing platform on meningitis," 2020 IEEE 2nd International Conference on Smart Cities and Communities (SCCIC), Ouagadougou, Burkina Faso, 2020, pp. 1-6.
22. A. Saha, M. N. Tasdid and M. R. Rahman, "Mining Semantic Web Based Ontological Data," 2018 21st International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2018, pp. 1-5.
23. D. Thakral and S. Ranjan, "Semantic Web based Recommendation System in E-Commerce Websites," 2022 International Conference on Smart and Sustainable

Technologies in Energy and Power Sectors (SSTEPS), Mahendragarh, India, 2022, pp. 218-222.

24. R. M. Mohammad and H. Y. AbuMansour, "An intelligent model for trustworthiness evaluation in semantic web applications," 2017 8th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2017, pp. 362-367.
25. A. C. S. Sheela and C. Jayakumar, "Comparative Study of Syntactic Search Engine and Semantic Search Engine: A Survey," 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 2019, pp. 1-4.
26. O. Nabuco, R. Bonacin and R. Martoglia, "Web2Touch 2017: Semantic Technologies in Smart Information Sharing and Web Collaboration," 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Poznan, Poland, 2017, pp. 275-277.
27. N. E. A. Amrani, O. E. K. Abra, M. Youssfi and O. Bouattane, "A new interpretation technique of traffic signs, based on Deep Learning and Semantic Web," 2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, 2019, pp. 1-6.
28. M. M. Martínez-González and M. -L. Alvite-Díez, "Thesauri and Semantic Web: Discussion of the Evolution of Thesauri Toward Their Integration With the Semantic Web," in IEEE Access, vol. 7, pp. 153151-153170.
29. W. Wu, Y. Jiang, C. Hao, Y. Shen and Y. Shen, "Fusion analysis of monitoring information points tables based on semantic Web and Hadoop technology," IEEE EUROCON 2017 -17th International Conference on Smart Technologies, Ohrid, Macedonia, 2017, pp. 142-145.
30. A. A. G. Y. Paramartha, K. Y. E. Aryanto and G. R. Dantes, "Integration of Region-based Open Data Using Semantic Web," 2018 Third International Conference on Informatics and Computing (ICIC), Palembang, Indonesia, 2018, pp. 1-6.

31. A. Mazayev, J. A. Martins and N. Correia, "Semantic web thing architecture," 2017 4th Experiment@International Conference (exp.at'17), Faro, Portugal, 2017, pp. 43-46.
32. P. Chhaya, Kyung-Hee Lee, Kwang-soo Shin, Chi-Hwan Choi, Wan-Sup Cho and Young-Sung Lee, "Using D2RQ and Ontop to publish relational database as Linked Data," 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, 2016, pp. 694-698.
33. D. Octaviani, A. Pranolo and S. Othman, "RDB2Onto: An approach for creating semantic metadata from relational educational data," 2015 International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 2015, pp. 137-140.
34. L. Yang, T. Xu and Z. Wang, "Agent based heterogeneous data integration and maintenance decision support for high-speed railway signal system," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 2014, pp. 1976-1981.
35. R. Devi, D. Mehrotra and H. Baazaoui-Zghal, "Pubworld-A R2rml Mapping Driven Approach To Transform Relational Database Data Into Shareable Format," 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 2018, pp. 221-227.
36. A. Julianita, S. Nugroho and B. W. Yohanes, "Mapping multiple databases to resource description framework with additional rules as conclusions drawer," 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, 2017, pp. 5-8.
37. K. U. Danyaro and M. S. Liew, "Semantic Web for Meteorological and Oceanographic Data," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 304-309.
38. D. Moldovan et al., "Tools for mapping ontologies to relational databases: A comparative evaluation," 2015 IEEE International Conference on Intelligent

Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2015, pp. 77-83.

39. S. Ramanujam, V. Khadilkar, L. Khan, S. Seida, M. Kantarcioglu and B. Thuraisingham, "Bi-directional Translation of Relational Data into Virtual RDF Stores," 2010 IEEE Fourth International Conference on Semantic Computing, Pittsburgh, PA, USA, 2010, pp. 268-276.
40. D. Moldovan, C. Pop, M. Antal, T. Cioara, I. Anghel and I. Salomie, "SWAG: Semantic web application generator - a library for using ontologies as web services," 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2016, pp. 103-110.
41. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин.
42. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».
43. Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу».
44. «ДСН 3.3.6.042-99 Санітарних норми мікроклімату виробничих при-міщень».
45. ДБН 13.2.5-28-2006 «Природне і штучне освітлення».
46. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».
47. ДСТУ 12.1.005-88 «ССБП. Загальні санітарно-гігієнічні вимоги до повітря робочої зони».
48. ДБН В.2.5-28-2006 «Інженерне обладнання будинків і споруд. При-родне і штучне освітлення».
49. ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Ви-моги до захисних заходів від ураження електричним струмом».

50. ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».
51. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, буди-нків та зовнішніх установок за вибухопожежною та пожежною небезпекою».
52. Прогнозування екологічних ризиків з використанням аналізу ієрархів та теорії нечітких множин: міжнародна науково-практична конференція «І-й всеукраїнський з'їзд екологів»: Тези доповідей. Україна, м. Вінниця, 4-7 жовтня 2016. – 2016. – С.25.
53. Клап Я. А., Яремкевич О. С., Червецова В. Г., Заярнюк Н. Л., Новіков В. П., Дослідження впливу електромагнітних, постійних магнітних та акустичних полів на організм людини // Вісник Нац. ун-ту "Львівська політехніка". – 2016 – № 812. – С. 365–372.
54. Сучасний стан досліджень впливу електромагнітних випромінювань на організм людини [Електронний ресурс]/[А. П. Чорний, В. В. Никифоров, Д. І. Родькін, В. Ю. Ноженко] // Інженерні та освітні технології в електротехнічних та комп'ютерних системах: щоквартальний науково-практичний журнал. – Кремен-чук: КрНУ, 2013.
55. Екологія та охорона навколишнього природного середовища: навч. посібник для вузів / В. С. Джигирей. - 6-те вид., випр. і доп. - К. : Знання, 2017. - 422 с.
56. Боротьба з шумом на виробництві: Довідник / Під ред. О. Я. Юдіна. – М: Машинобудування, 2015. – 297 с.