МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів аналізу безпеки семантичних баз даних
(тема)

Виконав:
Студент 2 курсу, групи ІПЗздм-19-1
Циблієва Н.О.
(прізвище, ініціали)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник проф. Шостак І.В.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« ____ » _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Циблієвої Ніни Олександрівни
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів аналізу безпеки семантичних баз даних

затверджена наказом університету від 26.03.2021 № 34 Стз

2. Термін подання роботи до екзаменаційної комісії 12 05 2021р.
3. Вихідні дані до роботи *проаналізувати існуючі алгоритми, що використовуються для вимог підтримки прийняття рішень, мови розробки програмного забезпечення*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, опис запропонованих варіантів оптимізації, використовувані методи та алгоритми, опис розробленої програмної системи, опис застосованих програмних рішень, аналіз можливих застосувань*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Мета завдання, обґрунтування доцільності розробки, постановка задачі, базові моделі, методи й алгоритми, структурно-логічна схема взаємодії даних, інтерфейс програмної системи, результати дослідної експлуатації програмної

системи, висновки

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
спецчастина	проф. Шостак І.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	26 березня 2021 р.	виконано
2.	Огляд існуючих методів	31 березня 2021 р.	виконано
3.	Розробка алгоритмів, проектування та розробка ПЗ	15 квітня 2021 р.	виконано
4.	Підготовка пояснювальної записки	28 квітня 2021 р.	виконано
5.	Спецчастина	30 квітня 2021 р.	виконано
6.	Підготовка презентації та доповіді	05 травня 2021 р.	виконано
7.	Попередній захист	10 травня 2021 р.	виконано
8.	Нормоконтроль, рецензування	10 травня 2021 р.	виконано
9.	Занесення роботи в електронний	11 травня 2021 р.	виконано
10.	Допуск до захисту в зав. кафедри	12 травня 2021 р.	виконано

Дата видачі завдання ____ _____ 2021р.

Студент _____
(підпис)

Керівник роботи _____ проф. Шостак І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ /ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра: 108 с, 5 табл., 46 рис., 6 дод., 33 джерела

СЕМАНТИЧНЕ ПАВУТІННЯ, СЕМАНТИЧНІ БАЗИ ДАНИХ, БЕЗПЕКА БАЗ ДАНИХ, СЕМАНТИЧНИЙ АНАЛІЗ.

Метою роботи є розробка погодженого набору алгоритмів контролю прямого доступу користувачів до елементів семантичних баз даних і контролю результатів логічних висновків, що дозволяють забезпечити комплексну безпеку семантичних баз даних.

Предмет досліджень – системи безпеки семантичних баз даних.

Методи дослідження – сучасні підходи до питань аналізу даних, безпеки баз даних.

Результат – розроблено алгоритми та архітектуру програмного забезпечення забезпечення безпеки семантичних БД.

SEMANTIC WEB, SEMANTIC DATABASES, DATABASE SECURITY, SEMANTIC ANALYSIS.

The aim of the work is to develop an agreed set of algorithms for controlling direct access of users to the elements of semantic databases and controlling the results of logical conclusions that allow to ensure the comprehensive security of semantic databases.

The subject of research – semantic database security systems.

Research methods – modern approaches to data analysis, database security.

The result – developed algorithms and software architecture for semantic database security.

Я, Циблієва Ніна Олександрівна, студентка гр. ІПЗздм-19-1, здобувачка вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів аналізу безпеки семантичних баз даних», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У наш час сучасні інформаційні системи організацій створюються на основі реляційних баз даних (БД), у яких в основному описується синтаксис даних. Використання реляційних БД має ряд недоліків, таких, як складність схем, недостатня виразність, складність інтеграції БД і відсутність можливості виконання логічних введення на даних. У зв'язку із цим починають створюватися й використовуватися інформаційні системи організацій, засновані на семантичних технологіях [1], основною ідеєю яких є перехід від роботи тільки із синтаксисом (структурою) інформації до роботи з її семантикою (змістом) і до семантичного моделювання (за допомогою онтологій) різних предметних областей. З обліком цього основним елементом семантичних інформаційних систем стають семантичні бази даних (СБД), які включають не тільки дані, але й семантичні моделі, на основі яких ці дані формуються. Для практичного використання подібних систем потрібно дослідити й розробити моделі й алгоритми для набору базових сервісів. Одним з таких сервісів є забезпечення безпеки роботи СБД. Задача даного сервісу полягає в тому, що тільки користувачі, що мають відповідні права, зможуть отримувати доступ до даних, що зберігаються в СБД, і вони не повинні мати якої-небудь можливості отримати недозволені їм елементи онтологій і метаданих за допомогою логічних правил.

У цей час розроблене багато методів забезпечення інформаційної безпеки операційних систем і реляційних баз даних. Але дані методи не можуть використовуватися для забезпечення безпеки семантичних БД. Це пов'язане з тим, що для СБД характерна сильна ієрархічна зв'язаність між елементами, а крім цього, у них є можливість отримання нової інформації на основі відомих фактів шляхом використання логічних правил. Для рішення задачі забезпечення безпеки СБД уже розроблені окремі методи й алгоритми, такі, як, наприклад: контроль доступу користувачів на основі іменованих RDF-графів, контроль доступу користувачів на рівні триплетів в RDF-сховищі [2]. Але розроблені методи й

алгоритми мають ряд недоліків, які не дозволяють ефективно забезпечити комплексну безпеку СБД.

Метою роботи є розробка погодженого набору алгоритмів контролю прямого доступу користувачів до елементів семантичних баз даних і контролю результатів логічних висновків, що дозволяють забезпечити комплексну безпеку семантичних баз даних.

Для досягнення поставленої мети потрібно розв'язати такі задачі:

– провести аналіз існуючих моделей, методів і алгоритмів забезпечення безпеки семантичних БД;

– розробити алгоритми формування погоджених рівнів безпеки всіх елементів онтологій, що зберігаються в СБД;

– створити алгоритм визначення рівнів безпеки всіх триплетів у семантичних БД;

– розробити алгоритм визначення рівнів безпеки результатів логічних висновків.

– розробити метод виявлення порушень безпеки результатів логічних висновків;

– розробити алгоритм контролю отриманих результатів при виконанні запитів до семантичних БД;

– створити архітектуру системи забезпечення безпеки семантичних БД;

– реалізувати програмне забезпечення підтримки безпеки роботи із семантичними БД.

Об'єктом дослідження є безпека семантичних БД при виконанні користувачами різних операцій.

Предметом дослідження є методи й алгоритми підтримки безпеки роботи із семантичними БД.

В впроцесі виконання кваліфікаційної роботи та розробці й дослідженні наступного набору нових алгоритмів забезпечення безпеки:

– розроблені алгоритми узгодження рівнів безпеки елементів семантичних БД, що відрізняються використанням принципу узгодження рівнів безпеки класів,

властивостей і індивідів;

– створений алгоритм визначення покриття безпеки семантичних БД, що відрізняється можливістю узгодження й визначення рівнів безпеки RDF-триплетів;

– запропонований алгоритм визначення покриття безпеки результатів логічних висновків семантичних БД, що відрізняється можливістю визначити рівні безпеки всіх результатів логічних висновків, отриманих шляхом використання логічних правил;

– створений метод виявлення порушень результатів логічних висновків у семантичних БД, що відрізняється можливістю представлення СБД у вигляді RDF-графів і контролю розкритих триплетів, на основі яких користувачі можуть отримати недозволені результати логічних висновків шляхом використання логічних правил;

– розроблений алгоритм контролю отриманих результатів при виконанні запитів до семантичних БД, що відрізняється керуванням відповідями на прямі й логічні запити.

В роботі потрібно створіти архітектуру ПЗ забезпечення безпеки семантичних БД, що дозволяє із припустимими затримками забезпечити підтримку безпеки СБД при інтенсивному навантаженні, і в можливості практичного використання розроблених методів і алгоритмів для підтримки безпеки роботи семантичних БД в інформаційних системах організацій. Запропоновані алгоритми узгодження рівнів безпеки елементів семантичних БД дозволяють надійно контролювати доступ користувачів до елементів онтологічних моделей, що зберігаються в семантичних БД.

1 АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

1.1 Аналіз технології Semantic Web

Сучасні інформаційні системи організацій створюються на основі реляційних БД. При проектуванні структур реляційних БД ураховується семантика даних, під якою розуміється зміст, що лежить у їхній основі, описуваний за допомогою взаємозв'язків між різними поняттями і їх властивостями. Після створення БД використовуються отримані схеми БД (таблиці й зв'язку між ними), а робота із семантикою даних уже не виконується. Однак у цей час в інформаційних системах усе більше потрібно працювати не тільки із синтаксисом даних, але й з їхньою семантикою. Використання семантики дозволяє підвищити якість роботи інформаційних систем за рахунок більш якісного опису ресурсів з використанням семантичних метаданих і виконання логічних висновків. Уся ця інформація зберігається в спеціальних БД, які називаються семантичними БД. Так само, як і для реляційних БД, потрібно забезпечити безпека даних, що зберігаються в семантичних БД.

Семантичні моделі й дані семантичних інформаційних систем описуються з використанням спеціальних мов, які входять до складу семантичних технологій Semantic Web [3].

У цей час для роботи із семантикою даних уже розроблені й продовжують розроблятися спеціальні семантичні технології [4], під якими розуміється набір стандартів і методів, що дозволяють описувати зміст даних (їх семантику) і виконувати роботу з ними.

Слова «семантичні технології» часто зустрічаються в описах концепції «Семантична Павутина» (Semantic Web), запропонованої [4] в 2001 році. Ціль Семантичної Павутини полягає в додаванні структурованої метаінформації до існуючих в WEB-мережі документів і даним для явного опису їх семантики, що дозволяє програмам виконувати більш якісну роботу із цими даними. В

подальшому в даній роботі під семантичними технологіями будуть розумітися семантичні технології концепції Semantic Web.

Семантичні технології складаються з багаторівневого набору різних стандартів і технологій, у яких кожний рівень використовує можливості нижчих рівнів.

Компоненти семантичних технологій для Semantic Web показано на рисунку 1.1 [2]:

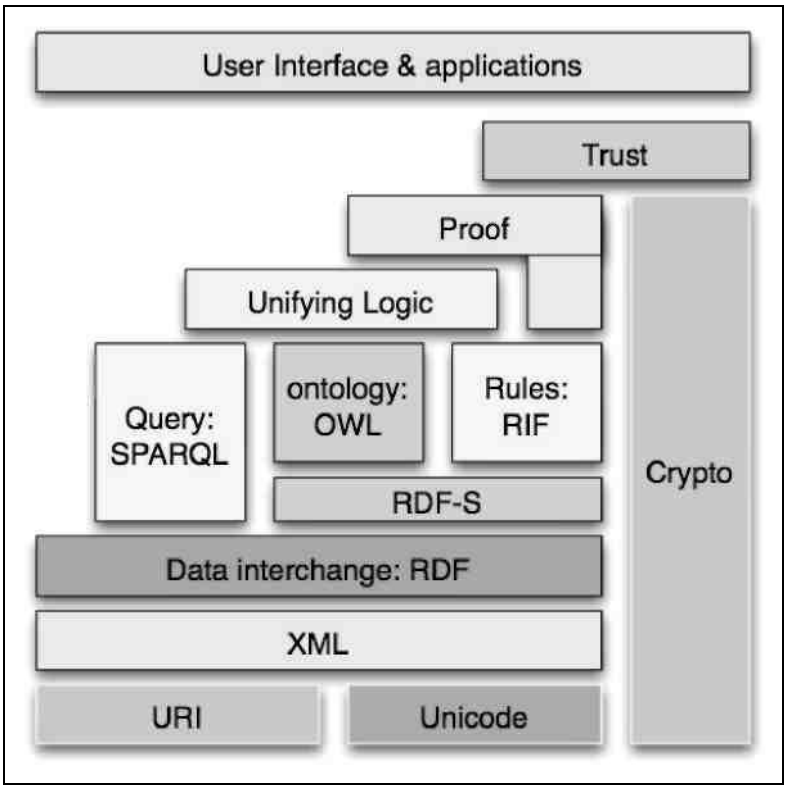


Рисунок 1.1 – Стек семантичних технологій Semantic Web [2]

– *UNICODE* – кодування, що визначає фундаментальний стандарт кодування даних;

– *URI* – спосіб і стандарт для завдання глобально унікальних ідентифікаторів ресурсів;

– *XML + NS + XMLSCHEMA* – метамова синтаксичного опису напівструктурованої інформації й пов'язані з ними стандарти (простору імен і схеми);

– *RDF* – модель даних для опису ресурсів і відносин між ними, може бути

серіалізована за допомогою мови *XML;*

– *RDFS* – мова для опису схем, що надає можливість створювати словники для опису RDF-даних;

– *RIF* – мова опису правил, що описує логічний висновок, виконуваний на семантичних даних [5];

– *OWL* – мова опису онтологій, що розширює можливості мови *RDFS;*

– *SPARQL* – протокол і мова опису запитів до RDF-даним;

– *Unifying Logic* – рівень, що дозволяє виконувати логічні висновки на семантичних даних;

– *ProOf* – рівень, що дозволяє доводити правильність виконання логічних висновків;

– *Trust* – рівень, що надає принципи й методи, що дозволяють досягати необхідного рівня довіри між взаємодіючими агентами.

1.2 Аналіз використання онтологій

Онтології почали використовуватися в області інформатики з 1980-х років дослідниками, що працюють в галузі штучного інтелекту. Спочатку вони використовувалися для обробки природної мови, а потім і для представлення знань. В 1990-х роках почалося дослідження можливості використовувати онтології для інтеграції й пошуку інформації в БД і мережі Інтернет. Пізніше онтології стають основними ключовими елементами, використовуваними для реалізації концепції семантичної веб-мережі.

Існують різні визначення онтології, одне з яких [6]: «Онтологія це формальний, точний опис (специфікація) погодженої концептуалізації». У даному визначенні термін «формальна» означає те, що онтологія є машиночитною структурою. Під терміном «погоджена концептуалізація» мається на увазі, що даний концептуальний опис не є чиєюсь приватною думкою, а думкою, з яким згодна

деяка група людей. А під терміном «концептуалізація» розуміється структура реальності, розглянута незалежно від словника предметної області й конкретної ситуації.

Онтологія включає модель (схему), що представляє собою опис множини понять і відносин між ними (онтологічна модель) і екземпляри понять.

Опис онтологій ґрунтується на формальних логіках. У якості таких логік використовується логіка опису [7]. Застосування логік дозволяє виконувати логічний висновок.

У цей час використовуються три способи класифікації онтологій: за ступеню формальності; за метою створення (призначенню) і за змістом. У свою чергу ці типи онтологій містять у собі подальшу класифікацію.

Ступінь формальності онтології відображає, як описується зміст онтології. Всі онтології можуть бути розділені за критерієм формальності на наступні групи: неформальні, напівформальні, напівформальні штучною мовою, формальні онтології.

У рамках класифікації онтологій по призначенню виділяють чотири рівня: онтології представлення, онтології верхнього рівня (наприклад, *CYC, DOLCE, SUMO),* онтології предметних галузей (наприклад, в області медицини створена велика кількість стандартних, структурованих словників, таких, як наприклад, *SNOMED-CT* і *UMLS)* і прикладні онтології.

Класифікація за змістом дуже схожа на класифікацію онтологій по призначенню, але в ній основна увага приділяється реальному змісту онтологій, а не абстрактної мети, переслідуваної авторами при їхньому створенні. Основними видами онтології за змістом є загальні онтології, онтології задач, предметні онтології.

На рисунку 1.2 показана загальна класифікація онтологій.

У цей час для опису онтологій консорціум Всесвітньої павутини (W3C) розробила спеціальну мову OWL – Ontology Web Language [8].

Мова OWL надає більше можливостей для опису семантики даних, раніше використовувані мови, такі, як OIL [9], DAML+OIL [10] і RDFS [11].

Рисунок 1.2 – Загальна класифікація онтологій

За ступеню виразності мова OWL ділиться на три частини:

– OWL Lite – для створення класифікаційних ієрархій і спрощення обмежень і є легко реалізованим;

– OWL DL (опис логіки) – підтримує максимальну виразність при збереженні обчислювальної повноти й можливості розв'язання;

– OWL Full – забезпечує максимальну виразність і синтаксичну волю RDF, але без гарантій обчислювальної здатності.

Мови OWL DL і OWL Lite розширюють словник RDFS, але накладають обмеження на використання цього словника для більш ефективної програмної обробки. Ці обмеження гарантують обчислювальну повноту й можливість розв'язання систем логічних висновків з використанням таких систем, як Fact++ і Pellet, які можуть виконувати логічний висновок в OWL-онтологіях на основі виразних описових логік (ОЛ).

Основними компонентами OWL-онтології є екземпляри, класи (поняття), атрибути й відносини [12].

Екземплярами (instances) або індивідами (individuals) є основні компоненти онтології, що перебувають на нижньому рівні. Вони можуть описувати як фізичні об'єкти (машина, університет, магазин), так і абстрактні (слова, числа). Між індивідами існують наступні відносини: два індивіду можуть бути представлені як той самий (sameas); індивід може бути представлений, що як відрізняється від інших індивідів (differentFrom); множина індивідів можуть бути представлені, що як взаємно відрізняються друг від друга {AllDifferent).

Індивіди в онтології можуть мати атрибути, що мають, принаймні, назву і значення, і які використовуються для зберігання інформації, специфічної для даного об'єкта й пов'язаної з ним.

Класами (class), або поняттями, є абстрактні групи, колекції або набори об'єктів. Між класами існують наступні відносини: є вбудований самий загальний клас за іменем Thing, який є класом усіх індивідів, і суперклас для всіх OWL-класів; класи можуть бути організовані в ієрархії за допомогою відносини rdfs:subClassOf; два класи можуть бути представлені як еквівалентні з використанням відносини equiva-lentClass. Клас може включати екземпляри, інші класи або їх комбінації.

У мові OWL властивості (відносини) розділяються на об'єктні властивості {owl:ObjectProperty) і властивості типів даних (owl:dataTypeProperty). Як правило, об'єктні властивості як об'єкт приймає тільки екземпляри класів або порожні вузли, а властивості типів даних – екземпляри типів даних (літерали, числа, рядки, дати й т.д.). Між властивостями існують наступні характеристики: одне відношення може бути презентовано як інверсія іншого (in-verseOf); відносини можуть бути оголошені транзитивними (TransitiveProperty), симетричними (SymmetricProperty), що мають унікальні значення (Function-alProperty), зворотньо-функціональними (InverseFunctionalProperty). Деякі інші відносини даної мови показано в табл. 1.1 [13].

Загальна структура основних елементів онтологій, описаних з використанням OWL, показано на рисунку 1.3.

Таблиця 1.1 – Відносини мови онтології

Характеристики відносин OWL-онтології	Правила висновку відповідного відношення
owl:inverseOf	$(p1, owl: inverseOf, p2) \wedge (s, p1, o) \rightarrow \{o, p2, s)$
owl:TransitiveProperty	$(p, rdf: type, owl: TransitiveProperty) \wedge (X, p, Y) \wedge (Y, p, Z$
owl:FunctionalProperty	$(p, rdf: type, owl: FunctionalProperty) \wedge (s, p, o) \rightarrow (s, p, o$ є єдиним для s
owl:InverseFunctionalProperty	$(p_1, owl: InverseFunctionalProperty, p_2) \wedge (s, p_1, o) \rightarrow (s, p_1$ і (o, p_1, s) є єдиними в одній базі даних
owl:sameAs	$(a, owl: sameAs, b) \rightarrow (b, owl: sameAs, a)$

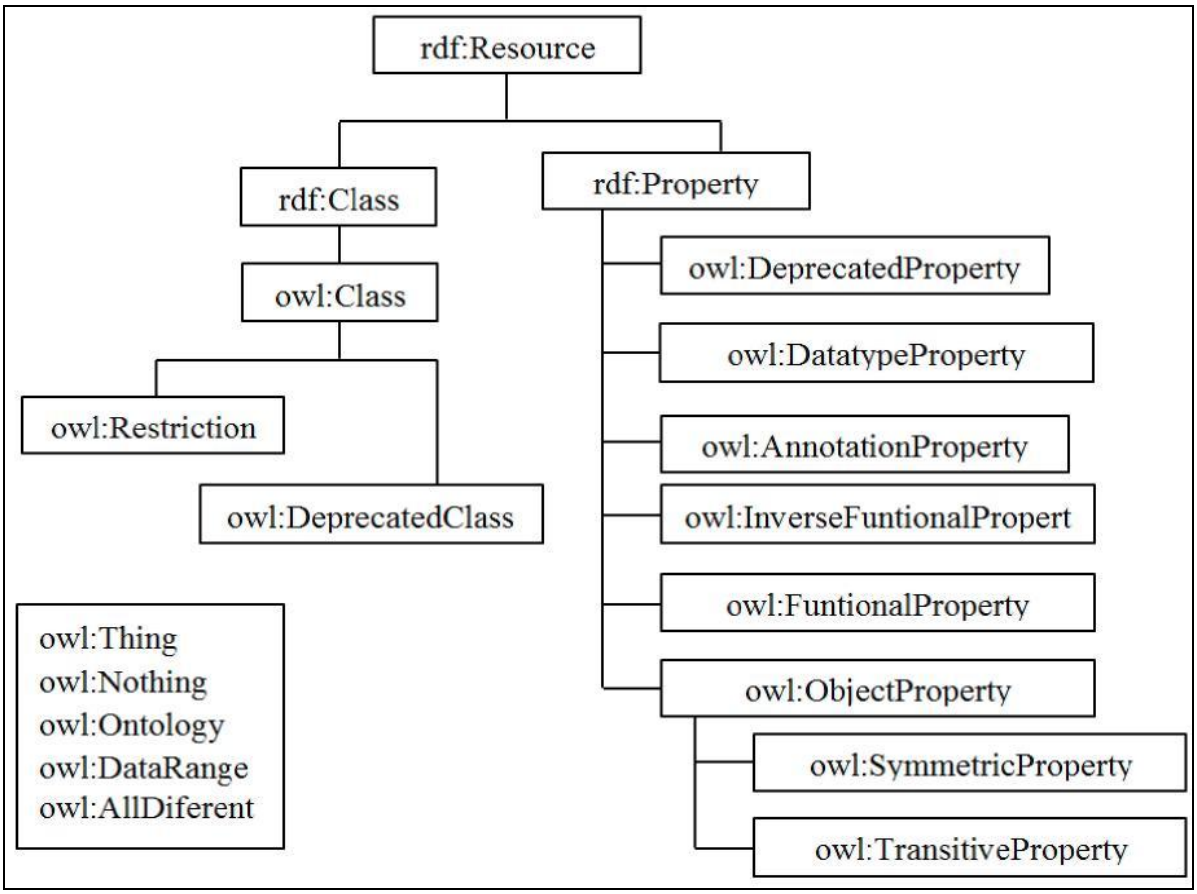


Рисунок 1.3 – Загальна структура основних елементів OWL онтологій [12]

У цей час існує багато програмних засобів, які мають можливість редагування, перегляду, імпорту онтологій різних форматів і мов; підтримують графічне редагування й керування бібліотеками онтологій. Прикладами таких

інструментів є такі системи, як Ontolingua [13], Protégé [14], OntoSaurus [15], OntoEdit , OilEd , WebOnto, WebODE.

Найбільше часто використовуваним є редактор онтологій Protégé. Це пов'язане з тим, що в порівнянні з іншими засобами він має наступні переваги:

– призначений для побудови (створення, редагування й перегляду) онтологій прикладної області. Його первісною метою є допомога розроблювачам програмного забезпечення в створенні й підтримці явних моделей предметної області й включення цих моделей безпосередньо в програмний код;

– дозволяє проектувати онтології, розвертаючи ієрархічну структуру абстрактних або конкретних предметних класів і слотів;

– структура онтологій зроблена аналогічно ієрархічній структурі каталогу. На основі сформованої онтології Protégé може генерувати форми отримання знань для введення екземплярів класів і підкласів;

– має графічний інтерфейс, зручний для використання недосвідченими користувачами, постачений довідками й прикладами;

– має набір плагінів, що дозволяє адаптувати його для редагування моделей, збережених у різних форматах (стандартний текстовий, у БД JDBC, UML, мов XML, XOL, SHOE, RDF і RDFS, DAML+OIL, OWL).

У загальному вигляді під метаданими розуміються структуровані дан характеристики, що представляють собою, даних для цілей ідентифікації, пошуку, оцінки й керування ними [16]. Вони описують об'єкти інформаційної системи й звичайно складаються з набору визначених елементів (властивостей або атрибутів), які описують різні аспекти об'єкта.

Структура метаданих може варіюватися, але завжди має наступні характеристики: кінцевий набір атрибутів; наявність назв в атрибутів; закріплений зміст кожного атрибута (трактування значення атрибута); можливість привласнити одному атрибуту кілька значень.

Будь-який об'єкт, описаний метаданими, може бути розглянутий у трьох різних аспектах: структура, контекст і контент.

Об'єкт характеризується як внутрішньою структурою, так і зовнішньої – відносинами з іншими об'єктами в інформаційній системі. Чим більше об'єкт структурований – внутрішньо й зовні, тем об'єкт простіше знаходити, простіше їм маніпулювати й визначати відповідність іншим об'єктам.

Контекст є зовнішнім стосовно об'єкта властивістю й визначається тим, хто, навіщо, коли і як створив цей об'єкт. Контекст дозволяє ідентифікувати об'єкт серед множини інших об'єктів.

Об'єкт створюється в інформаційній системі для надання користувачам необхідної інформації, і ця інформація передається через інформаційний зміст об'єкта – контент [17].

Метадані можуть описувати один або більш аспектів об'єкта із трьох наведених вище. У рамках семантичних технологій найбільша увага приділяється дослідженню таких метаданих, які описують контекст і контент об'єкта.

Семантичні метадані – це метадані, що описують контекст і/або контент об'єкта в інформаційній системі за допомогою понять предметної області, на деяких мов опису онтології [18].

За допомогою семантичних метаданих може бути усунута лексична багатозначність термінів, використаний для опису інформаційних об'єктів. Крім цього, за допомогою онтологій може бути визначена відповідність між різними інформаційними об'єктами.

Для опису метаданих W3C розроблена спеціальна мова опису ресурсів RDF – Resource Description Framework.

Мова RDF є машиночитуваною мовою, що дозволяють виконувати кодування, обмін і повторне використання структурованих метаданих між різними програмами [19]. Такі можливості, наслідувані від метамови XML [20], підвищують корисність семантичної інформації [21], [22].

Перевагою мови RDF перед метамовою XML є можливість представлення інформації, що міститься в ресурсах, у вигляді орієнтованого графа, у той час як мова XML дозволяє описувати тільки ієрархічні структури.

RDF використовує триплети для опису ресурсу і його відносини з іншими ресурсами в графові. RDF-триплет складається із трьох елементів:

– суб'єкт (*subject*): URI-ідентифікатор, або порожній вузол;

– предикат (*predicate*): URI-ідентифікатор;

– об'єкт (*object*): URI-ідентифікатор, або літерал.

Триплет позначається як $t = [s, p, o]$ де s – суб'єкт, p – предикат, o – об'єкт. Множина RDF-триплетів утворює орієнтований граф, у якому вершинами є суб'єкти й об'єкти, а ребра позначені предикатами.

На рисунку 1.5 показаний опис простого висловлення: ресурс *rec:employee* має значення *Hoangquyen* і зв'язаний відношенням *rec:know* з ресурсом *rec:director*. У цьому випадку літерал *Hoangquyen* і ресурс *rec:director* є об'єктами.

Із цього прикладу важко зрозуміти, що являють собою ресурси *rec:employee* і *rec:director* і які інші відносини можуть бути застосовані для їхнього опису. Людина може здогадатися, що ці ресурси є людськими, виходячи з того, що вони беруть участь у відношенні `rec:know`. Такий же логічний висновок може зробити й програма, що використовує мову опису схем або онтологій, наприклад такий, як RDFS (див. рис. 1.5).

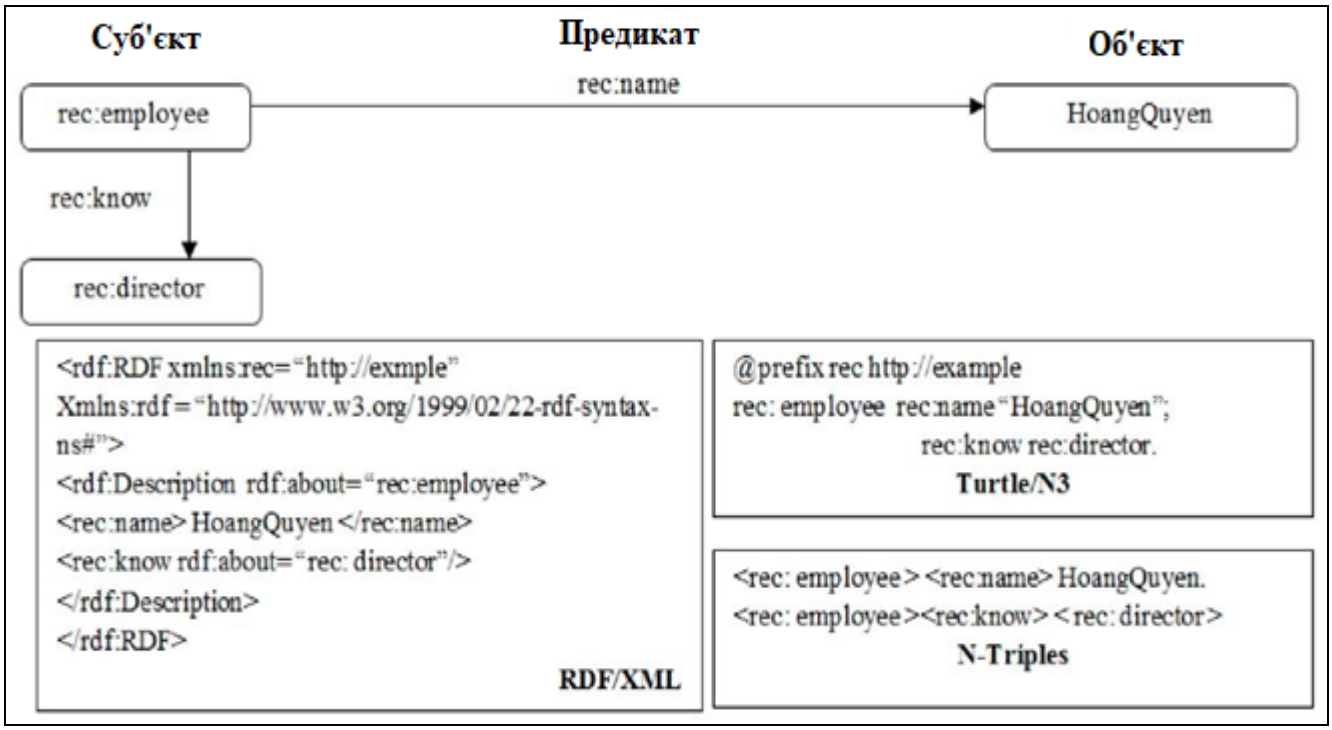


Рисунок 1.5 – RDF-триплети й формати серіалізації

RDF-дані можуть записуватися у різних форматах, наприклад таких, як *N3*, *N-Triples*, *Turtle*, *RDF/XML* [23]. З них найбільш зручними для людини є формати *N3* і *Turtle*.

1.3 Логічні мови в семантичних технологіях

Описи онтологій ґрунтуються на різних логіках. До таких логік відноситься логіка предикатів першого порядку [24], фреймова логіка й дескриптивна логіка. Логіка предикатів першого порядку в мові Ontolingua [19], фреймова логіка в мові F-Logic, а описова логіка (ОЛ) використовується в таких мовах, як DAML-ONT, OIL, DAML+OIL і OWL [25].

Для опису логічного висновку в мові опису онтологій OWL використовується описова логіка [7] у зв'язку з такими її властивостями, як можливість розв'язання; виразна варіативність і автоматична класифікація понять.

Властивість можливості розв'язання є важливим тому, що при використанні формалізму в рамках програмної системи не повинне виникати ситуацій, коли отримання відповіді від системи логічного висновку неможливо, отже, неможливо й виконання операцій, заснованих на логічному висновку. Можливість розв'язання логічної мови гарантує отримання відповіді. Але залежно від обчислювальної складності логічної мови на пошук відповіді може бути витрачена різна кількість часу. Існує пряма залежність між виразною потужністю логічної мови і його обчислювальної ресурсоємності. Чим виразніше мова, тим більше точно можна описати предметну область за допомогою цього мова, але й час, витрачене на логічний висновок, буде значним. Описова логіка (ОЛ) дозволяє знайти компроміс між виразними потребами й доступними обчислювальними ресурсами. ОЛ представляє множина логічних мов, що володіють різною виразною потужністю. Залежно від задачі можна вибрати мову з достатньою

виразністю й мінімальної обчислювальної ресурсоємності. У цьому полягає властивість виразної варіативності дескриптивної логіки [26].

Властивість автоматичної класифікації понять засноване на логічному висновку й гарантує, що для кожного поняття буде визначене місце в ієрархії понять (таксономії) виходячи з опису поняття. Ця властивість використовується для побудови таксономії понять, на основі якої розроблені методи семантичної обробки інформації.

У описових логіках синтаксичними будівельними блоками є атомарні поняття (унарні предикати), атомарні ролі (бінарні предикати) і представники (індивіди, константи). Ролі (властивості, відносини) є самостійними елементами, які потім можуть бути пов'язані з поняттями.

Поняття й ролі можна поєднувати у вираз для опису більш складних понять за допомогою конструкторів (операцій). Між поняттями й між ролями можна задати відносини (які поняття (ролі) є тотожними і які поняття (ролі) включають інші поняття (ролі)).

Виразна міць мови обмежується тим, що вона використовує досить малий набір конструкторів для побудови складних понять і ролей. Неявні знання [27] про поняття й індивідів можуть бути виведено з явних [28] автоматично за допомогою процедур висновку. Зокрема, важливу роль відіграють відносини включення (родо-видові відносини – *subsumption relationships*) між поняттями й відносинами екземплярів.

Мова Semantic Web Rule Language SWRL (*мова опису правил у Семантичній Павутині*) [29] дозволяє включати правила в опис онтологій OWL-DL, що заснований на об'єднанні мов OWL і Ruleml.

Особливості його використання для опису логічних правил полягають у наступному:

– правила SWRL не містять конкретних об'єктів, а тільки посилаються на них, що дає можливість застосовувати те саме правило для декілька груп об'єктів;

– правила SWRL можуть бути додані до OWL-опису, тобто включені в онтологію;

– складання й читання правил зручніше виконувати, якщо для цього існує спеціальна мова.

Можливість роботи з SWRL реалізована в багатьох редакторах онтологій, таких, як Protégé і Ontolingua. Він підтримується такими фреймворками, як Jess, Sesame і Jena [30].

Найпоширенішою мовою запитів до семантичних даних є мова SPARQL (Semantic protocol and RDF query language) розроблений організацією W3C. У порівнянні з іншими мовами, SPARQL має наступні переваги [31]:

– використовується для представлення запитів до різноманітних джерел даних незалежно від того, зберігаються ці дані безпосередньо в RDF або представляються у вигляді RDF за допомогою проміжного програмного забезпечення.

– має можливості формування запитів до обов'язкових і необов'язкових графовим шаблонам разом з них кон'юнкціями й диз'юнкціями.

– підтримує тестування розширеного значення й обмеження запитів за допомогою вихідного RDF-графа;

– результати запитів SPARQL можуть бути представлені у вигляді результуючих наборів або RDF-графів.

Більша частина запитів SPARQL включає набір шаблонів триплетів, який називається основним графовим шаблоном. Шаблони триплетів подібні RDF-триплетам, за винятком того, що кожний суб'єкт, предикат і об'єкт може бути змінної. Основний графовий шаблон відповідає підграфу RDF-даних, отже, RDF-терміни даного підграфа можуть бути замінені на змінні, а результатом також є RDF-граф.

1.4 Аналіз семантичних баз даних

Під семантичної БД (СБД) розуміється база даних, у якій зберігаються онтології, семантичні метадані й множина логічних правил. Вона може бути описана, як $DB_S = \{O, M, R\}$, де O – онтологія, M – семантичні метадані, R – множина логічних правил.

Множина логічних правил R використовується для отримання логічних висновків на основі відомих даних. Дана множина може бути розділене на дві підмножини $R = \{R_1, R_2\}$, де R_1 – множина онтологічних логічних правил, включених у мові опису онтологій; R_2 – множина користувацьких логічних правил, створених розроблювачами й/або користувачами для отримання можливих логічних висновків.

Основні елементи семантичних БД показано на рисунку 1.6.

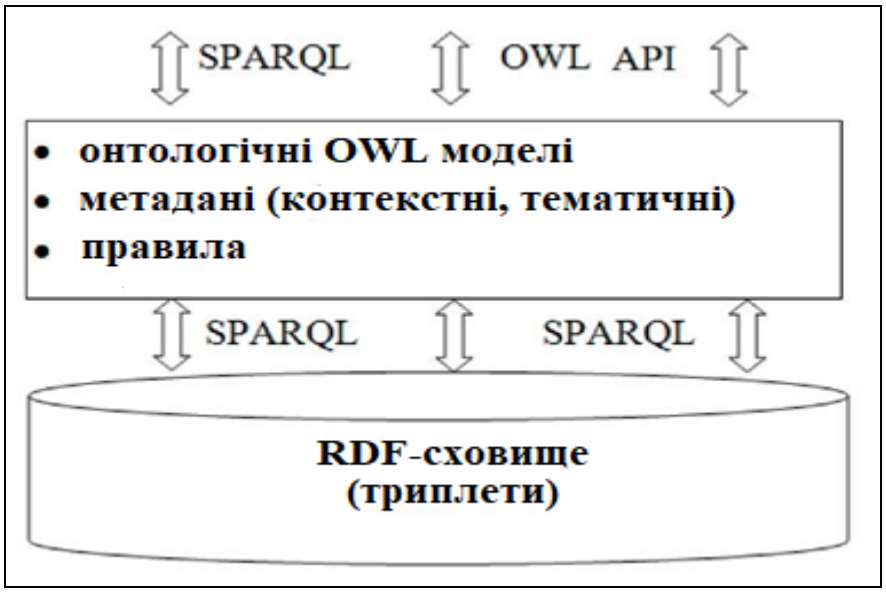


Рисунок 1.6 – Структура семантичної бази даних

У семантичних БД онтології, метадані представляються у вигляді множин RDF-триплетів, що зберігаються в RDF-сховищах (RDF stores).

Семантичні БД будуються на основі RDF-сховищ. Під RDF-сховищем (RDF-stores, RDF-triple store) розуміється інформаційна підсистема, призначена для зберігання й надання доступу до RDF-даним (триплетам). По своїй архітектурі RDF-сховища діляться на два типи: реальні RDF-сховища (native stores) і RDF-сховища, підтримувані реляційними СУБД (DBMS-backed stores) [32].

Реальні RDF-сховища (native stores) повністю реалізують ядро БД, яке оптимізовано для обробки RDF-даних і працює незалежно від будь-якої іншої системи керування базою даних (СУБД). У цих сховищах дані зберігаються безпосередньо у файловій системі, в одному файлі або розділяються на кілька файлів.

RDF-сховища, підтримувані реляційними СУБД (DBMS-backed stores), використовуються для виконання зберігання й пошуку даних існуючих СУБД.

Різняться два види моделі зберігання: загальна схема та схема конкретної онтології.

Загальна схема RDF-сховища може складатися з різних таблиць для зберігання триплетів, наприклад таблиці, що полягає із трьох стовпців: суб'єкта, предиката й об'єкта. При цьому необхідно дотримувати принципів нормалізації схеми такий БД шляхом зберігання всіх URI-ідентифікаторів і літералів в окремих таблицях. Такий підхід з використанням загальної схеми має гнучкість представлення схеми зберігання, однак вимагає об'єднання великої кількості таблиць для відповіді на складні запити.

Підхід на основі схеми конкретної онтології полягає в зберіганні триплетів у наборі спеціальних таблиць, структура яких відбиває структурні властивості конкретної онтології. Можна виділити три різні види схем:

– для кожного класу онтології (горизонтальне представлення) створюється окрема таблиця, кожний стовпець якої відповідає одній властивості класу, а кожний рядок – одному екземпляру класу. Основним недоліком такого підходу є необхідність реструктуризації таблиць БД при зміні структури онтології;

– для кожного предиката створюється окрема таблиця (вертикальне представлення), що містить два стовпця: для суб'єкта й об'єкта. При використанні такого підходу значно знижується ефективність виконання складних запитів у зв'язку з необхідністю об'єднань таблиць, відповідних до властивостей, використовуваних у даному запиті;

– гібридні схеми – поєднує переваги обох попередніх підходів, при цьому кожний клас представляється у вигляді таблиці, яка містить тільки ідентифікатори екземплярів даного класу.

Відомими реальними RDF-сховищами (native stores) є такі системи, як: 4/5 store, AllegroGraph, OWLIM, а прикладами RDF-сховищ, заснованих на реляційних СУБД, є наступні: 3store, Jena SDB, Oracle llg, а також гібридні сховища – Redstore, Sesame, Virtuoso, BigData [33].

Під системою керування семантичними БД розуміється сукупність програмних засобів, що забезпечують керування створенням і використанням даних у таких БД. Системи керування семантичними БД мають наступні можливості:

– організація зберігання RDF-даних;

– надання програмного інтерфейсу для витягу інформації зі збережених RDF-даних за допомогою мови структурованих запитів SPARQL або спеціального інтерфейсу програмованого додатків (application programming interface – API);

– підтримка функцій адміністрування збережених даних: додавання, видалення, модифікація й розподіл прав доступу.

У наш час існує багато різних систем керування семантичними БД, такі, як Redland, FreeBase, Sesame, Oracle 11g Release, Virtuoso Universal Server [22].

Семантична СУБД Sesame використовується для зберігання й виконання запитів до наборів RDF-даних. Вона створена з використанням мови Java і підтримує наступні можливості:

– універсальний користувацький інтерфейс API;

– сховище для RDF-даних;

– функції адміністрування для роботи з RDF і OWL-даними;

– середовище розробки й виконання Java;

– виконання логічних висновків для RDF-даних;

– наявність вбудованого WEB-сервісу, називаного кінцевої SPARQL-крапкою (SPARQL Endpoint) виконання, що реалізує протокол, SPARQL-запитів і підтримуючого стандарт серіалізації результатів SPARQL-запитів [22].

Під Oracle 11g Release розуміється об'єктно-реляційна система керування БД компанії Oracle, у якій підтримується підсистема керування семантичними БД, яка має наступні можливості:

– надання інтерфейсу API для роботи із клієнтськими додатками;

– підтримка сховищ для RDF-триплетів;

– підтримка функцій адміністрування даних, збережених для роботи з

даними в RDF-форматах, таких як RDF / XML, N-triple і Turtle;

– виконання SPARQL-рапитів і Rdql-запитів до RDF-даних;

– включення високопродуктивної системи логічного висновку для наступних предикатів: *owl.sameAs, rdfs:subClassOf і rdfs:subPropertyOf;*

– підтримка роботи з мовами Perl, PHP, Python і Ruby.

Virtuoso Universal Server (VUS) є однієї з найефективніших СУБД, у яку включається підсистема керування семантичними БД. У порівнянні з іншими СУБД у ній підтримується можливість виконання логічних висновків для OWL-онтологій. Основними її можливостями є наступні:

– універсальний користувацький інтерфейс;

– підтримка сховищ Quad-based (дані зберігаються у вигляді квадів – кортежів із чотирьох елементів <граф, суб'єкт, предикат, об'єкт>);

– підтримка функцій адміністрування даних, збережених для роботи з даними в RDF-форматах, таких, як RDF / XML, N-triple і Turtle;

– включення високопродуктивної системи логічного висновку з динамічною матеріалізацією, що обробляє наступні предикати: *owl.sameAs, owl:equivalentClass, owl:equivalentProperty, owl:Inversefmct-ionalProperty, rdfs:subClassOf, rdfs:subPropertyOf, owl:inverseOf, owl.SymmetricProperty* і *owl:TransitiveProperty;*

1.5 Інформаційні системи на основі семантичних баз даних

На основі семантичних БД і технологій розробляються нові різновиди інформаційних систем. Загальна архітектура семантичної інформаційної системи [34] логічно розділена на 5 рівнів (див. рис. 1.7). У даній архітектурі:

– відповідно до потоків керування й даних компоненти більш високого рівня використовують і запитують дані в компонентів більш низьких рівнів, і одна операція компонентів на більш високому рівні може запустити на виконання

кілька операцій на більш низькому рівні;

– у відповідності за ступенем абстрактності вхідних компонентів у них компоненти з однаковим рівнем абстрагування включені в той самий рівень.

Рівень джерел даних включає всі види джерел даних, такі, як семантичні БД, бази даних, файли різного формату, WEB-сервіси, будь-які зовнішні онтології, доступні за URI, і т.д.

Джерела даних розглядаються компонентами даного рівня, тому що робота з ними може виконуватися компонентам і більш високого рівня.

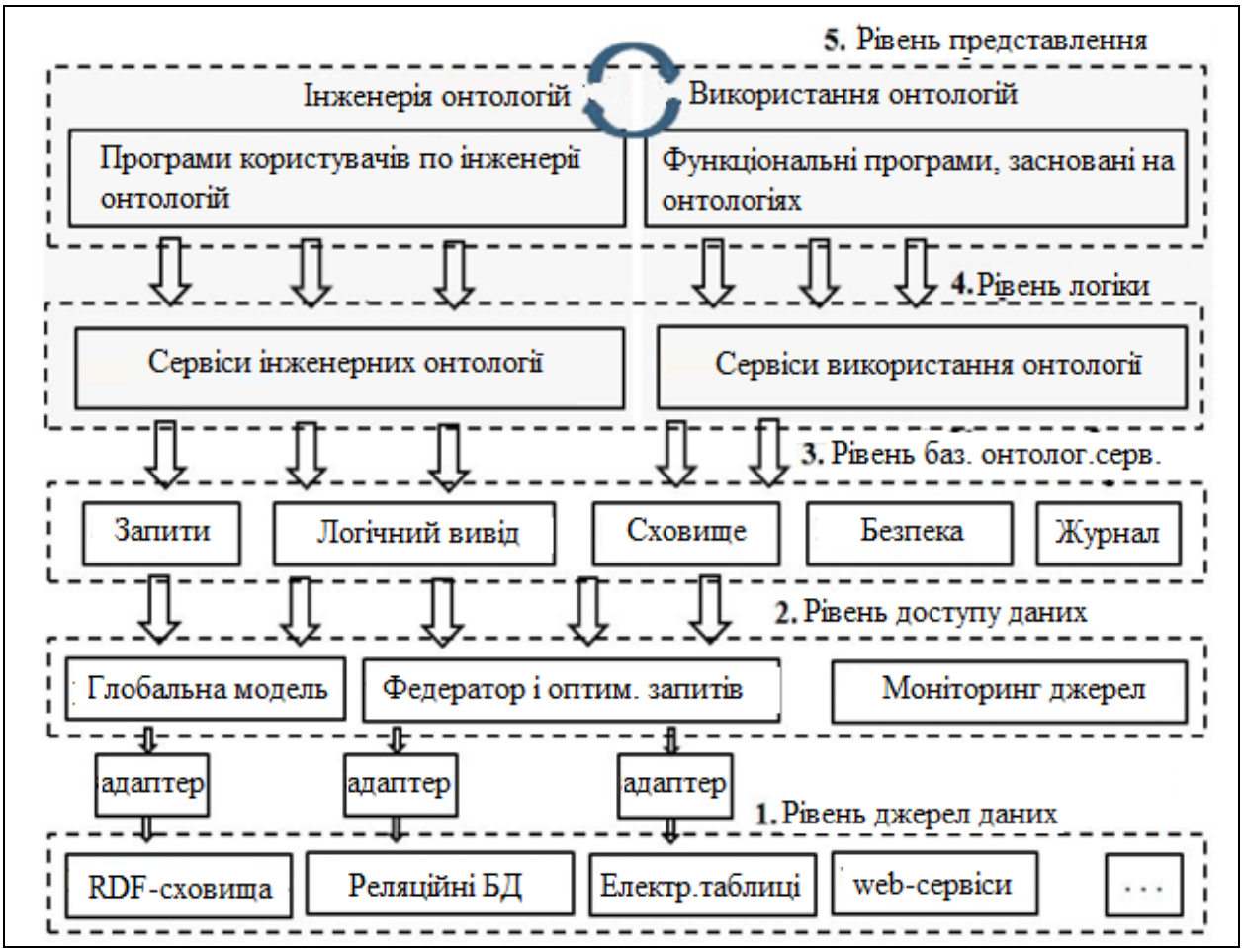


Рисунок 1.7 – Загальна структура інформаційної системи на основі семантичної бази даних

На рівні доступу до даних виконується абстрагування, що дозволяє сховати особливості реалізації й відмінність форматів.

Об'єктні моделі даного рівня можуть включати дані, що надходять із таких джерел, як звичайні джерела даних (найбільше часто це реляційні БД) або

джерела семантичних даних (онтологічних джерел). Даний рівень надає API і специфічні адаптери. Під адаптерами джерел даних розуміються системи, що дозволяють у реальному часі виконувати перетворення даних з вихідних форматів в RDF-модель.

Рівень базових онтологічних сервісів містить у собі наступні компоненти:

– сервіси реєстрації онтологій (публікація, пошук);

– сервіси роботи з онтологіями, такі, як отримання й зміна елементів;

– сервіси обробки запитів на отримання даних;

– сервіси логічного висновку;

– сервіси ведення журналу роботи з онтологіями;

– сервіси підтримки безпеки семантичних БД.

Сервіси підтримки безпеки семантичних БД відіграють більшу роль у процесі роботи семантичних інформаційних систем. Вони повинні мати наступні можливості:

– забезпечення доступу користувачів до даних, що зберігаються в сховищі;

– контролю виконання запитів до БД;

– забезпечення логічних висновків, отриманих при виконанні відправлених запитів.

1.6 Постановка задач дослідження

У цей час розроблений досить повний набір засобів роботи із семантикою інформації, таких, як: RDF – мова опису ресурсів, OWL – мова опису онтологій, SPARQL – мова запитів до семантичних БД, SWRL – мова опису логічних правил.

Зберігання семантичної інформації може бути реалізоване за допомогою семантичних БД. У цей час уже розроблені такі системи керування семантичної БД, як наприклад: Sesame, Oracle 11g Release, Virtuoso Universal Server.

На основі семантичних БД активно створюються інформаційні системи,

такі, як наприклад: семантичні інформаційні портали й електронні бібліотеки.

При роботі із семантичними БД потрібно розв'язати дві основні проблеми: контроль доступу користувачів до даних і контроль результатів логічних висновків.

Існуючі підходи забезпечення безпеки СБД не мають усі необхідні функції для вирішення даних проблем.

У даній роботі пропонується система підтримки безпеки роботи із семантичними БД, що володіє всіма перерахованими вище можливостями.

Дана система розробляється на основі моделей контролю доступу користувачів до даних і контролю результатів логічних висновків.

Для реалізації пропонованої системи забезпечення безпеки семантичних БД необхідно розв'язати набір задач, таких, як узгодження рівнів безпеки елементів онтології, визначення рівнів безпеки триплетів і результатів логічних висновків, виявлення порушень результатів логічних висновків і контроль отриманих результатів при виконанні запитів.

Модель контролю доступу користувачів створюється на основі наступних алгоритмів:

– визначення рівнів безпеки елементів онтологій і метаданих;

– визначення покриття безпеки (рівнів безпеки всіх триплетів) у семантичних БД;

– застосування дискреційної й мандатної політики безпеки.

Модель контролю результатів логічних висновків у СБД створюється на основі наступних методів і алгоритмів:

– визначення рівнів безпеки всіх результатів логічних висновків у СБД;

– визначення можливості отримання результатів логічних висновків між елементами;

– виявлення порушень результатів логічних висновків у СБД.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Аналіз об’єктних моделей

Рівень логіки включає сервіс інженерії онтологій і сервіс виконання базових операцій з онтологіями й метаданими (екземплярами понять), які специфічні для варіантів використання (включають логікові обробки) і працюють із конкретними об'єктними моделями (з конкретними даними). Дані сервіси викликають сервіси життєвого циклу онтологій *(ontology lifecycle services)* для керування семантичними даними й виконання вибірок.

Об'єктні моделі на даному рівні можуть включати дані, що надходять із різних джерел, таких, як звичайні джерела даних (наприклад, реляційні БД), джерела семантичних даних (онтологічні джерела).

Рівень представлення включає компоненти, що мають графічний інтерфейс користувача, такі, як WEB-додатки й desktop-застосунки, взаємодіючи з компонентами рівня логіки, з якими користувачі працюють. Запити до сервісів виконуються на рівні логіки при роботі з додатками цього рівня. Дані додатки діляться на дві групи:

– додатки підтримки інженерії онтологій, що включають у себе редактори онтологій, браузери онтологій і ведення бібліотек онтологій;

– додатки, що використовують онтології, що включають у себе портали, електронні бібліотеки, спеціальні інформаційні системи, системи прийняття керуючих рішень; системи керування знаннями.

Прикладами таких інформаційних систем є: Expert locator service (сервіс по пошукових експертам, організація NASA); Targeted drug assessment (цільова оцінка ліків, компанія Eli Lilly); Intelligent automobile diagnostics (інтелектуальна діагностика автомобілів, компанія Renault) [4]; A drug compound knowledgebase (інтегрована база знань по лікарським засобам, компанії Pfizer); Mobile content search and discovery (мобільний пошук документів, компанія Vodaphone) [5].

Семантичний інформаційний портал розроблений для керування базами знань [33]. На рисунку 2.1 показана загальна архітектура семантичного інформаційного порталу.

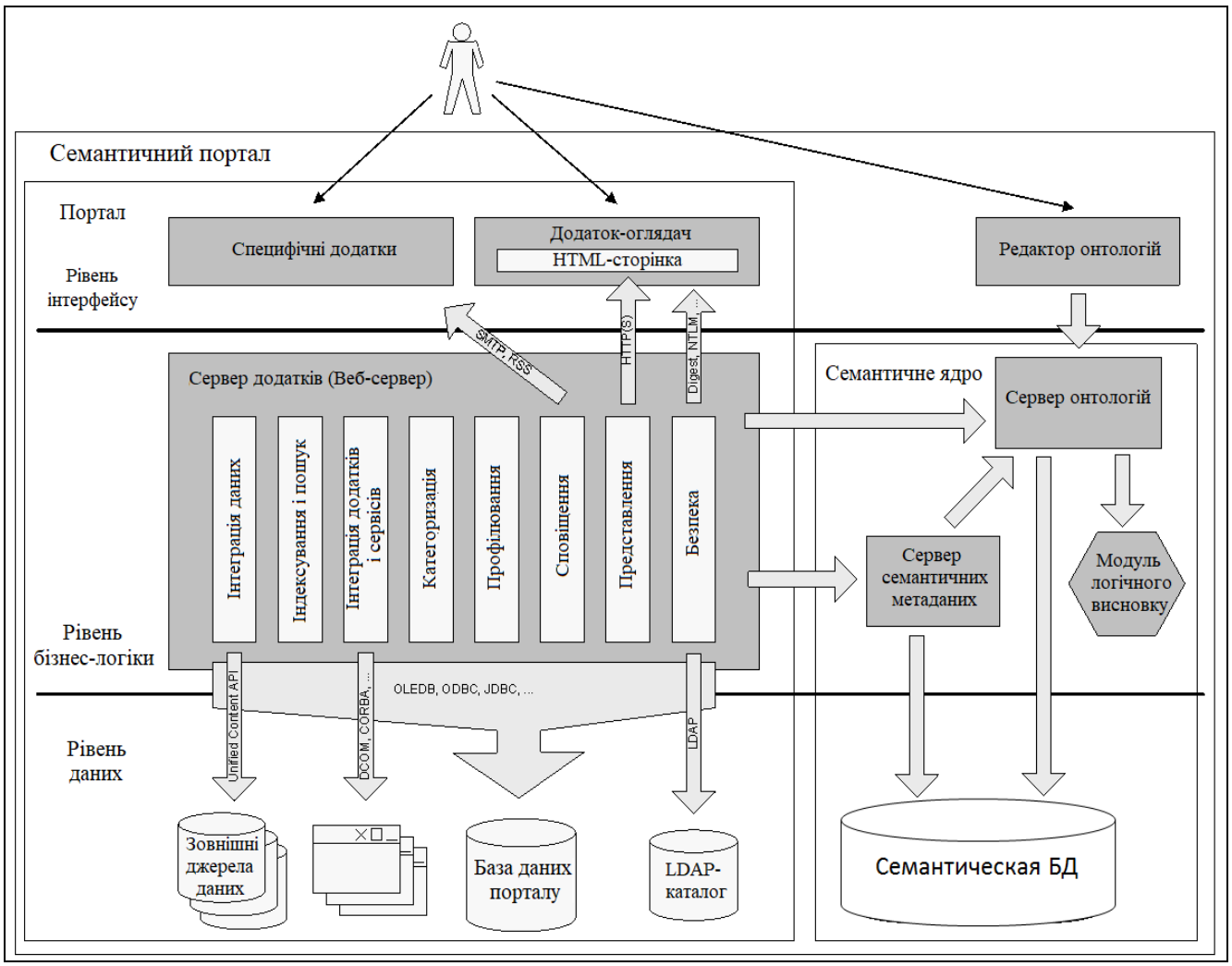


Рисунок 2.1 – Семантичний інформаційний портал

На рівні інтерфейсу користувача використовується тонкий клієнт (WEB-браузер), здатний візуалізувати представлення інформації, описане на мові HTML. Для використання деяких функціональних можливостей порталу користувач може використовувати й деякі інші клієнтські додатки (наприклад, клієнт електронної пошти, RSS-клієнт і т.д.).

На рівні бізнес-логіки використовується множина модулів, таких, як інтеграція даних, індексування й пошук, інтеграція додатків і сервісів, категоризація, профілювання, оповіщення, представлення й безпека. На цьому рівні дані інтегруються з різних джерел і зберігаються в семантичній базі даних.

Модуль безпеки виконує функції ідентифікації користувачів при підключенні до порталу й формування контексту безпеки при їхній роботі з функціями порталу.

На рівні даних різні БД використовуються для зберігання інформації з різних джерел. Семантична БД використовується для зберігання семантичних онтологій і метаданих у вигляді RDF-триплетів.

2.2 Проблеми забезпечення безпеки семантичних баз даних

У зв'язку зі створенням інформаційних систем комерційних і державних організацій на основі семантичних технологій дуже важливої стає проблема захисту даних від несанкціонованого використання. Усе більш актуальної стає задача контролю доступу до даних і їх використання з урахуванням заданих прав користувачів. При відсутності ефективних методів і алгоритмів контролю доступу користувачів до семантичних даних порушується їхня безпека, що може привести до небажаних наслідків [35].

У цей час був розроблений набір моделей, методів і алгоритмів забезпечення безпеки операційних систем і реляційних баз даних, але вони не можуть бути застосовані для семантичних БД, тому що в СБД є сильна ієрархічна зв'язаність між елементами й можливість отримання нової інформації користувачами на основі відомих фактів шляхом використання логічних правил [3].

Існують різні методи й алгоритми забезпечення безпеки семантичних БД, але вони включають тільки деякі з необхідних функціональних можливостей:

– контроль доступу користувачів до окремих елементів онтологій;

– контроль доступу користувачів до триплетів і їх компонентам (суб'єктові, предикату, об'єкту);

– контроль доступу користувачів до RDF-графам у СБД.

Для забезпечення надійної безпеки семантичних БД необхідно розробити модель контролю доступу користувачів до них, яка одночасно має всі перераховані можливості.

Крім цього, як уже було відзначено, у семантичних БД на основі відомих даних за допомогою логічних правил можна отримувати результати логічних висновків [24]. З використанням цього користувачі *U* можуть отримувати інформацію, до якої вони не мають права доступу. Внаслідок цього, актуальної є задача контролю результатів логічних висновків у семантичних БД. У цей час існує метод контролю логічних висновків у семантичних БД шляхом контролю доступу користувачів до логічних правил [25]. Даний метод не гарантує, що користувачі будуть отримувати результати логічних висновків відповідно до їхніх прав доступу.

Таким чином, можна виділити наступні дві основні групи задач, які необхідно розв'язати для забезпечення безпеки семантичних БД:

– контроль доступу користувачів до окремих елементів СБД;

– контроль результатів логічних висновків у СБД.

У цей час уже відомі наступні методи, моделі й системи контролю доступу користувачів до СБД:

– підсистема безпеки в сховище BigData [25], створена на основі моделі контролю доступу користувачів до іменованих RDF-графам;

– модель AC4RDF, розроблена на основі методів контролю доступу користувачів на рівні триплетів RDF-сховища;

– підсистема безпеки Allegrograph [26], розроблена на основі фільтрів безпеки;

– система RAP (Policy-Based Access Control for an RDF Store), створена на основі політики контролю доступу до RDF-сховища;

– методи контролю доступу користувачів до онтології [26].

– контроль логічних правил [26].

У моделі безпеки RDF-сховища на рівні RDF-графів контроль доступу користувачів до даних RDF-сховища виконується в такий спосіб:

– усі триплети збираються в набори триплетів, які називаються іменованими графами;

– кожному іменованому графові задаються рівень безпеки;

– кожному користувачу задаються роль і права доступу;

– користувач U може мати доступ і виконати різні операції над триплетами відповідно до політики безпеки, заданої іменованому графові, якому ці триплети належать.

Дана модель має високу ефективність у випадку, коли в кожному іменованому графові згрупована більша група триплетів. Однак якщо в іменованому графові є тільки одне або два твердження, то використовується модель «statement level provenance» [27], яка дозволяє визначити походження кожного триплету за допомогою SPARQL-запитів, у такий спосіб можна реалізувати політикові безпеки для триплетів. Дана модель використана для забезпечення безпеки даних в RDF-сховище BigData.

Модель Access Control for RDF stores (AC4RDF) реалізує контроль доступу користувачів на рівні триплетів RDF-сховища. Дана модель використовується для забезпечення безпеки RDF-сховища Sesame. Це виконується шляхом перевірки прав користувачів, у результаті якої визначається, хто має права доступу до RDF-триплету, що зберігається в RDF-сховище. У даній моделі, права доступу описуються власником RDF-даних за допомогою редактора Policyeditor [27], який дозволяє задати доступ користувачів до кожного RDF-твердженню або до графа RDF-даних, які зберігаються в RDF-сховище.

Загальну архітектуру системи AC4RDF показано на рисунку 2.2.

При відправленні користувачами U запиту q до RDF-сховищу модуль Access Control знаходить інформацію про обліковий запис користувачів і використовує модуль «політик Protune» [28] для вибору політики, що застосовується для даного запиту користувачів.

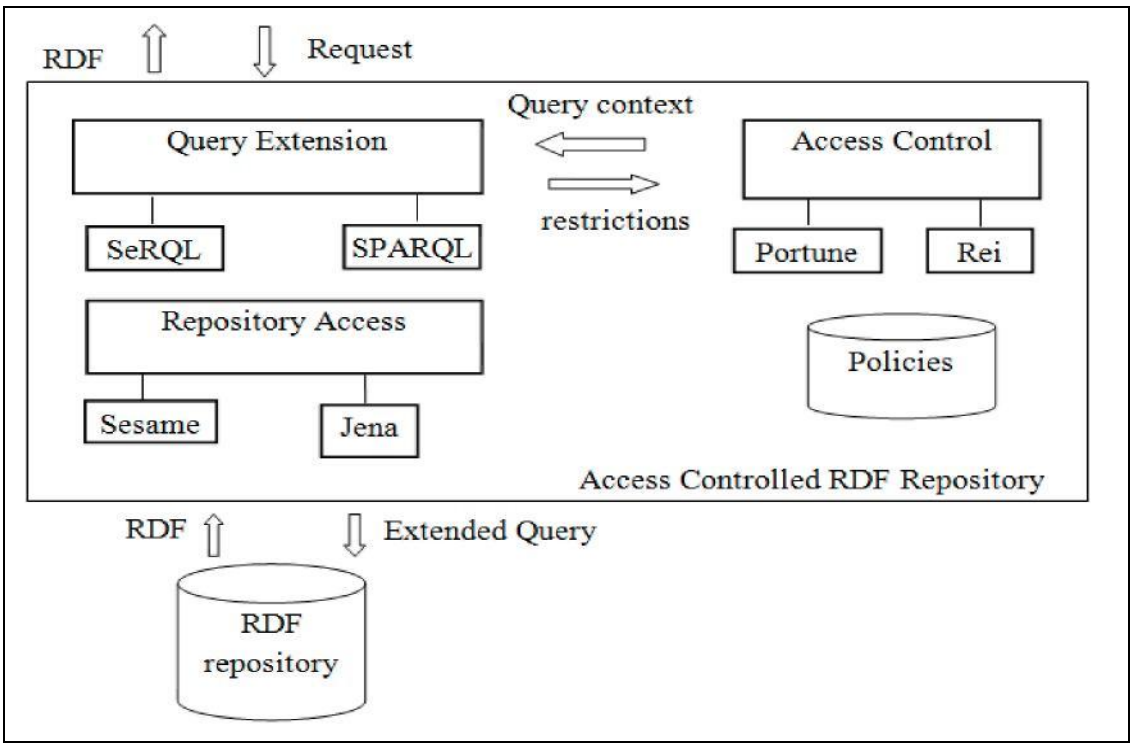


Рисунок 2.2 – Загальна архітектура системи AC4RDF

Модуль Rei перезаписує запит відповідно до певної політики. Переписаний запит відправляється в RDF-сховище й користувачі *U* можуть отримувати відповіді на даний запит (див. рис. 2.3).

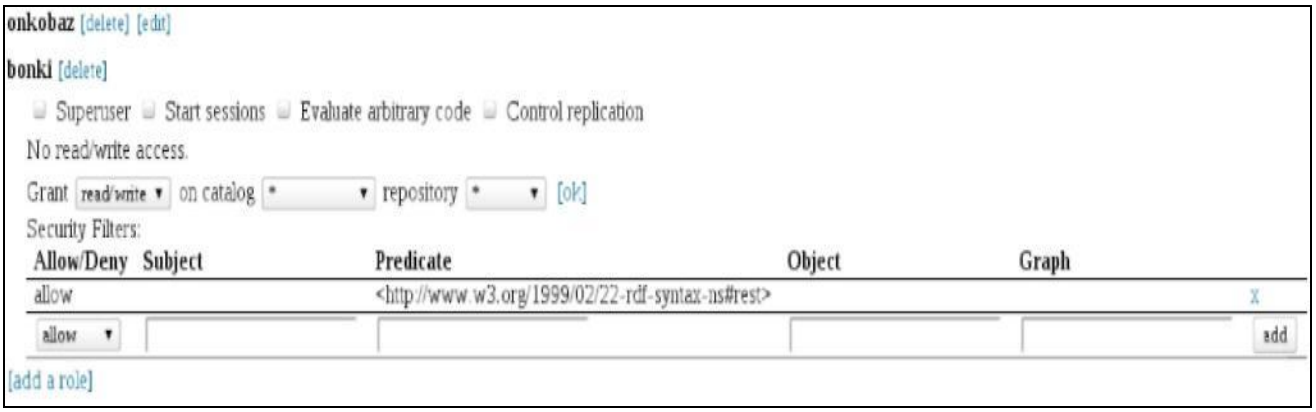


Рисунок 2.3 – Графічний інтерфейс для створення фільтра безпеки користувачів

У семантичній БД Allegrograph 4.11 для контролю доступу користувачів до RDF-сховищам застосовується підсистема безпеки, побудована на основі фільтра безпеки (filter secrutiry) [24], який створюється адміністратором сховища.

Адміністратор має всі права для керування даними й створення прав доступу для реєстрованих користувачів. Користувачеві задається роль, значення якої вибирається з множини {*Superuser, Start sessions, Evaluate arbitrary code,*

Control replication} і права – з множин {читання, запис, модифікація або видалення}. За фільтром безпеки адміністратор задає користувачам права доступу до яких-небудь сховищ, категорій даних. Крім цього, користувачі *U* можуть мати доступ тільки до конкретного триплету або до всіх триплетів, у яких міститься конкретний предикат, суб'єкт або об'єкт.

Приклад політики безпеки: користувачі *U* мають права на перегляд усіх триплетів, у яких міститься предикат *rec:Sarary*.

У процесі роботи із триплетами в RDF-сховище користувач *U* може вилучити або додати основні триплети, які є елементами онтологій або загальної схеми, отже, структура схеми (онтології) даних порушена. Для рішення даної проблеми була розроблена система контролю доступу користувачів до RDF-сховищу, заснована на політиках, що визначають права доступу користувачів [88].

Усі дії користувачів над сховищем проходять через модуль політики системи RAP, щоб визначити дія «дозволене» або «заборонене». У системі RAP усі триплети метаданих і політики доступу до них зберігаються в самому RDF-сховище (див. рис. 2.4).

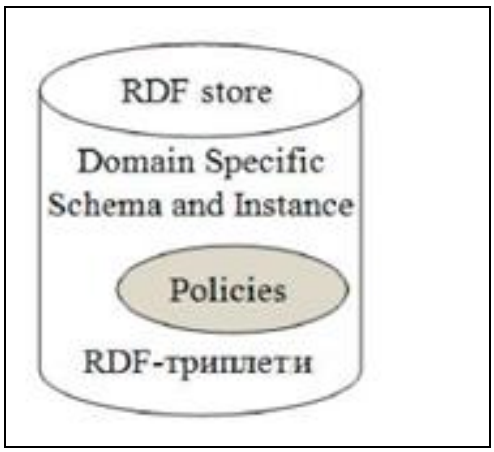


Рисунок 2.4 – Дані в RDF-сховищі

Система RAP побудована на основі фреймворка Jena, у якій підтримує засіб аналізу й виконання простого логічного висновку на RDF, RDFS і OWL. Політики системи RAP задаються як правила, які використовуються в її онтології для роботи із засобами логічного висновку RETE [36]. Загальна архітектура даної системи показано на рисунку 2.5.

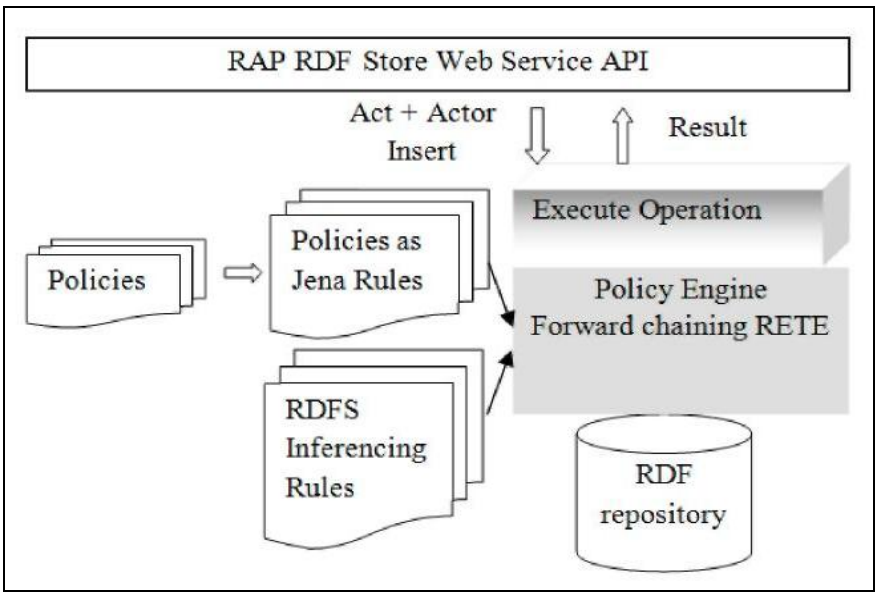


Рисунок 2.5 – Архітектура системи RAP

Система RAP підтримує виконання користувачами різних операцій, таких, як додавання, видалення й модифікація RDF-триплетів відповідно до їхніх прав доступу й з коректністю схеми даних в RDF-сховище.

Проблема забезпечення безпеки онтології згідно схем політики безпеки для керування доступом до понять онтологій і їх екземплярам створюються рівні безпеки, а користувачам – рівні доступу.

Керування доступом користувачів до онтологій виконане шляхом порівняння рівнів безпеки понять із рівнями доступу користувачів. Якщо рівень доступу користувачів більше рівнів безпеки понять, то користувачі мають доступ до понять онтології, отже, вони можуть мати доступ до всіх екземплярів даних понять.

Дана система може виконувати контроль тільки на рівні понять онтології, але не розуміє семантику й відносини між елементами онтологій.

Система контролю доступу користувачів до окремих елементів онтології, побудована на основі підходу CLP (constraint logic programming – логічне програмування в обмеженнях) – створена модель, у якій міститься схеми онтологій і семантичних даних. Дані в цій моделі представляються у вигляді RDF-дерева, на основі якого виконуються всі операції, що дозволяють контролювати доступ користувачів до елементів онтології.

Крім перерахованих вище методів і систем контролю доступу користувачів до онтології й RDF-сховищам також існують і інші методи, описані в [30].

У цей час також запропоновані й різні методи контролю доступу до логічних правил [30] – в основному всі вони засновані на використання рівнів доступу для логічних правил. У загальному вигляді вони можуть бути описані в такий спосіб: Нехай

$$DB_S = \{ O, M, R\},$$

де O – онтології;

M – семантичні метадані;

$R = \{r_1 , ... , r_n\}$ – множина логічних правил.

Основні можливості й обмеження розглянутих вище підсистем, моделей і методів забезпечення безпеки показано в таблиці 2.1.

Таблиця 2.1 – Особливості й обмеження підсистем, моделей і методів забезпечення безпеки СБД

Підсистеми, методи, автори	Основні функції	Недоліки
Підсистема безпеки в RDF-сховище BigData	Керування доступом на рівні іменованих RDF-графів	Ні можливості контролю доступу до триплетів і їх компонентам
AC4RDF	Керування доступом користувачів до RDF-триплету	Ні можливості контролю доступу до окремих елементів
Підсистема безпеки Allegrograph	Керування доступом до конкретного триплету або до всіх триплетів, у яких міститься конкретний предикат, суб'єкт або об'єкт	Система не розуміє семантики БД. Ні можливості контролю результатів логічних висновків
Система RAP	Політика доступу зберігаються в RDF-сховище. Керування доступом на рівні триплетів	Ні можливості контролю доступу до окремих елементів. Ні можливості контролю результатів логічних висновків
Підсистема L. Qin, V. Atluri	Контроль доступу до понять онтології та їх екземплярам	Ні можливості контролю доступу до атрибутів і відносин онтології

Продовження табл. 2.1

Підсистема N. Yialelis, E. Lupu, M. Sloman	Контроль доступу користувачів до окремих груп елементів онтології на основі RDF-дерева	Ні можливості контролю результатів логічних висновків
Контроль логічних висновків	Контроль доступу до логічних правил	Ні можливості виявлення порушень результатів логічних висновків при виконанні логічних правил

Тоді використовується наступна політика безпеки семантичних БД логічні правила, що включають:

– визначається множина рівнів безпеки $SL = \{sl_1 ..., sl_k\}$;

– кожному користувачеві U задається рівень доступу $sl_U \in SL$ для виконання логічних правил;

– кожному логічному правилу $r_i \in R$ задається рівень доступу $sl_{ri} \in SL$;

– якщо $sl_U \geq sl_{ri}$, то користувач U може виконувати логічне правила $\mathbf{r_i}$; інакше він це правило використовувати не може.

Даний метод дозволяє користувачеві U виконувати логічні правила відповідно до його рівня доступу, але не гарантує, що він буде отримувати результати відповідно до його прав доступу. Це пов'язане з тим, що в семантичних БД даним можуть бути задані рівні безпеки, які перевищують рівень доступу користувачів до правил sl_U.

У результаті їх аналізу можна зробити висновок про те, що не існує системи забезпечення безпеки семантичних БД, яка має наступні функціональні можливості:

– контролем доступу користувачів до триплетів і їх компонентам (суб'єктові, предикату, об'єкту);

– контролем доступу користувачів до RDF-графам у СБД;

– контролем результатів логічних висновків, отриманих користувачами шляхом використання логічних правил.

3 АРХІТЕКТУРА СИСТЕМИ БЕЗПЕКИ СЕМАНТИЧНИХ БАЗ ДАНИХ

3.1 Онтологічні моделі в семантичній базі даних

До основного змісту СБД відносяться: метадані й онтологічні моделі різних предметних областей, які використовуються для опису різних інформаційних ресурсів, а також логічні правила, за допомогою яких користувачі можуть отримувати результати логічних висновків.

Семантична БД визначається як $DB_S = \{O, M, R\}$, де O – єдина онтологічна модель ресурсів, M – семантичні метадані, R – множина логічних правил. Під єдиною онтологічною моделлю ресурсів [32] розуміється набір взаємозалежних онтологій, призначених для реалізації основних функцій СБД: $O = \{O_б, O_п\}$, де $O_б$ – базові онтології СБД, а $O_п = \{O_1, ..., Om\}$ – онтології основних предметних галузей знань для опису змісту електронних ресурсів [33]. До базових онтологій $O_б$, наприклад, відносяться: онтологія компанії, онтологія виробів, онтологія користувачів, онтологія ресурсів і онтологія системи і т.д. А до $O_п = \{O_1, ..., Om\}$ ставиться ієрархічно організована, послідовно розширювана система онтологій основних галузей знань для опису змісту даних у СБД [34].

Будь-яка онтологія O_i може бути формально описана в такий спосіб:

Під онтологією O розуміється знакова система (C, P, E, F, L, AC), $C = \{c_1, ..., c_m\}$ – кінцева множина елементів, які називаються поняттями (класами); $P = \{p_1, ..., p_n\}$ – множина елементів, які називаються властивостями (двомісні предикати); E – частковий порядок на множинах C і P, що задає відносини «підклас» і «суперклас»; F – функція, яка призначає кожному елементу множини P множина елементів із C (з обліком їх ієрархії в E), до яких воно застосовне (область дії, *domain)* і множина елементів з множини C, або літералів (екземплярів примітивних типів, таких, як рядка й числа), які можуть бути їхніми значеннями (область можливих значень, *range*); $L = \{Lc, LP, a_c, a_p\}$ – множина текстових міток Lc, LP для понять і відносин, які визначають професійні терміни, використовувані в ресурсах бібліотеки і їх відповідність: a_c – елементи множини C, a_p – елементи

множини *P; AC* – набір аксіом онтології – тверджень про елементи предметної області, які вважаються вірними, виражені з використанням відповідної логічної мови.

Таким чином, модель онтології містить:

– множину понять (класів) C, для кожного з яких визначаються властивості;

– строкові літеральні значення *(L_C)* – тестові мітки для їхньої інтерпретації (терміни, коментар і ін. на якій-небудь природній мові), наприклад: співробітник, документ і т.д.;

– URI-ідентифікатор поняття (URI – уніфікований ідентифікатор ресурсів), заданий за якимись правилами;

– необов'язкова початкова множина його екземплярів.

Множина відносин (предикатів) *P,* для кожного з яких задаються:

– URI-ідентифікатор предиката;

– строкові літеральні значення *(L_P);*

– область дії *(domain)* і область значення *(range),* задані за допомогою *F;*

– необов'язкові можливі властивості з наступних властивостей: функціональне, транзитивне, рефлексивне, симетричне, асиметричне.

Множина часткових порядків ***E***, яке називається відношенням успадкування *C x CuP x P.*

У єдиній онтологічній моделі СБД всі поняття й відносини, що входять у множину *CuP,* вважаються спадкоємцями універсального поняття *Thing,* яке розглядається в якості суперкласу всіх понять і відносин онтологій.

У семантичних БД логічним правилом $r_i \in R$ є вираз, який позначається як $\forall x_1, ..., x_m (b_1 \wedge ... \wedge b_k) \rightarrow a$, де $k \geq 1$ и $x_1, ..., x_m$ – це вільні змінні в $b_1 \wedge ... \wedge b_k$. Кожний b_i являє собою твердження (триплет), що має вигляд $[\alpha, \beta, \gamma]$, де α, β, γ – змінні, константи або OWL-аксіоми. Ліва частина правила $(b_1 \wedge ... \wedge b_k)$ називається тілом, а права частина правила *(a)* – головою правила. При складанні

таких правил необхідно, щоб усі змінні, що включаються в голову, містилися в складі тіла правила.

Нехай $\forall x_1, ..., x_m (b_1 \wedge ... \wedge b_k) \rightarrow a$ є логічним правилом $r_i \in R$, де кожний b_j (j = 1, ..., *k*) і *a* являють собою триплети. Тоді триплет $r_{Li} = [\alpha, \beta, \gamma]$ є *результатом логічних висновків* при виконанні логічного правила pp у семантичній БД, якщо задовольняє наступним умовам: існує функція відображення *f*, яка задовольняє наступним умовам:

f зберігає всі константи (*f(constant) = constant);*

$f([x_1, y_1, z_1]) = [\alpha_1, \beta_1, \gamma_1]$ и $f([x_1, y_2, z_2]) = [\alpha_2, \beta_2, \gamma_2]$, то $\alpha_1 = \alpha_2$;

$\forall f(bi) \in M, (i = 1, ..., k);$

$f(a) = r_{Li}$.

При роботі з DB_S для отримання результатів логічних висновків *RL* на основі відомих даних використовується множина логічних правил $R = \{R_1, R_2\}$, де $R_1 = \{r_{11}, ..., r_{1n}\}$ – множина логічних правил *гі,* які описуються на основі відносин між елементами онтологій; $R_2 = \{r_{2l}, ..., r_{2m}\}$ – множина логічних правил r_{2i}, створених користувачами/розробниками за допомогою мови SWRL.

До логічних правил $R_1 = \{r_{11}, ..., r_{1n}\}$ відносяться наступні елементи онтологій:

– характеристики властивостей онтології: симетричність, інверсія, транзитивність, функціональність, назад-функціональність.

– обмеження властивостей: *owl:allValuesFrom, owl.someValuesFrom, owl:cardinality* (кардинальність), *owl:hasValue.*

– картування онтологій: еквівалентність між класами й властивостями *(owl:equivalentClass, rdfs:subClassOf equivalentProperty),* ідентичність між індивідами *(owl.sameAs),* різниця індивідів *{differentFrom, allBiffereni).*

– прості властивості: *ObjectProperty*, *DatatypeProperty*, *rdfs.subPropertyOf rdfs:domain, rdfs:range.*

Логічні правила для характеристик властивостей онтології *O* можуть бути описані в такий спосіб:

owl.SymmetricProperty: (s, $p_u o$*)* (про, p_u *s).*

owl.-InverseOf {p_u owl:inverseOf p_2) л (s,p_u o) ->(про,*p_2, s).*

owl.-TransitiveProperty:

(p, rdftype, owl: TransitiveProperty) A (X, p, Y) A (Y, p, Z) (X, p, Z).

owl.FunctionalProperty: (p, rdftype, owl: FundionalProperty) A (s, p, o) → (s, P, o).

owl.PnverseFunctionalProperty:

(ph owl: InverseFunctionalProperty, p_2) A (S, pu o) (s, p_1, o) i (o, p_1, s).

До множини логічних правил R_2 відносяться більш складні логічні правила, які не можуть бути описані логікою онтологій. Множина R_2 описане користувачами або розроблювачами для отримання результатів логічних висновків *RL* залежно від множини відносин $P \in O$ й від предметної області, описуваною даною онтологією.

Наприклад, для отримання результатів логічних висновків відносини «виробник» використовується логічне правило:

(*?x є виконавцем ?y*) (*?z є продуктом ?y*) → (*?x є виробником ?z*).

А для отримання результатів логічних висновків відносини «виконавцем» використовується логічне правило:

(*?x є персоною в ?y*) (*?x є автором ?z*) (*?z описує ?g*) → (*?y є виконавцем ?g*).

Показані в прикладах логічні правила описані мовою SWRL. Вони не можуть бути отримані за допомогою мови опису онтологій OWL.

3.2 Засоби забезпечення безпеки семантичних баз даних

Для кожного користувача *U* семантичної БД у системі підтримки безпеки створюється обліковий запис, що містить відомості, які користувач *U* повідомляє про себе системі забезпечення безпеки. Основними елементами облікового запису користувача є ім'я користувача й пароль доступу. Значення пароля доступу зберігається в зашифрованому або хешованому вигляді [11] для забезпечення його безпеки.

Обліковий запис може містити також додаткові дані, що описують інформацію про користувачів, такі, як ім'я, прізвище, по батькові, стать, дата народження, e-mail адреса, домашня адреса, робоча адреса, номер домашнього телефону й т.ін. Обліковий запис користувачів зберігається в самій семантичній БД. Кожний користувач *U* відноситься до деякої групи або має деяку роль по роботі зі СБД.

Група – це іменована сукупність користувачів. Об'єднання користувачів у групу полегшує адміністрування СБД і, як правило, будується на основі формальної або фактичної структури організації. Наприклад, множина груп може бути наступним: {розроблювачі програми, менеджменти проекту, директор компанії} і т.д.

Роль – це заздалегідь певна сукупність правил, що встановлюють припустиме взаємодію між користувачами інформацією, що й захищається.

Під правами доступу розуміється сукупність правил доступу до інформації, що захищається, установлених правовими документами або власником, власником інформації.

Приклад взаємозв'язку ролей і прав доступу користувачів при роботі із семантичними БД показано в таблиці 3.1.

Таблиця 3.1 – Ролі й права доступу користувачів у СБД

Роль	Права доступу
Власник	Перегляд, оновлення, видалення й додавання даних
Адміністратор	Керування обліковими записами користувачів у СБД. Перегляд, видалення й додавання даних
Звичайний користувач	Перегляд, оновлення даних
Гості	Перегляд даних

Власник може переглянути, вилучити, додати будь-які свої дані в СБД.

Адміністратор може переглянути, вилучити, додати будь-які дані в СБД, а також він може управляти обліковими записами користувачів у СБД.

Звичайний користувач може обновляти й переглянути дані.

Гість тільки має право на перегляд даних.

Права доступу користувачів U можуть задаватися в семантичній БД різними способами, які залежать від використовуваної в ній політики безпеки.

Під політикою безпеки керування доступом (security policy access control) розуміється сукупність правил керування доступом користувачів до інформації, що захищається (даним).

У цей час існує багато політик безпеки. Найбільш відомими є дискреційна, мандатна й рольова політики безпеки [34]. На основі даних політик створюється система забезпечення безпеки роботи із семантичними БД, у якій політика безпеки ґрунтується на наступних правилах:

– існує набір ролей U_r і прав доступу U_p;

– створюється множина міток рівнів безпеки *MS*;

– усім даним в DB_S задаються рівні безпеки sl_D;

– кожному користувачеві *U* завдається рівень доступу sl_U;

– власник може вказати рівень безпеки sl_D для своїх даних.

– користувач має доступ до даних тоді й тільки коли його задані права доступу до даних дискреційною політикою або $sl_U \geq sl_D$.

Для визначення можливості отримання доступу користувачів до даних СБД виконується порівняння рівня доступу користувачів sl_U з рівнями безпеки даних sl_D, що зберігаються в СБД.

Під рівнем безпеки sl_D даних розуміється рівень захищеності (таємності) даних, який визначає, у якому ступені дані доступні для користувачів. Наприклад, дані можуть бути доступними всім користувачам або бути зроблене секретними, до яких доступ має тільки дуже обмежена група користувачів.

Значення sl_D задається з множини міток рівнів безпеки даних *MS*, яке може бути презентовано у вигляді набору номерів, рядків і т.д. Ієрархічні відносини між різними елементами *MS*, (що задаються символами "<", "≤", "≥" або ">") дозволяють описати ступінь захищеності даних. Наприклад, множина міток рівнів безпеки може бути наступною:

MS = {«абсолютно секретно», «секретно», «конфіденційно», «несекретно»}, де «абсолютно секретно» > «секретно» > «конфіденційно» > «несекретно».

Для різних систем значення елементів множини *MS* може різнитися.

Наприклад, *MS* може виглядати в такий спосіб, де кожний елемент *MS* є словом:

MS = {« абсолютно секретно», «секретно», «конфіденційно», «несекретно»}.

Кожний елемент *MS* є номером: *MS* = {0, 1... 100} і т.д.

Відповідно до рівнів безпеки даних кожному користувачеві *U* задається рівень доступу sl_U, що дозволяє йому мати доступ до даних.

Вважається, що якщо $sl_U \geq sl_D$, то користувач *U* має доступ до даних *D*, інакше він не має доступу до цих даних.

У семантичних БД для забезпечення безпеки кожному елементу онтології *O* и кожному триплету *t* семантичних метаданих *M* можуть задаватися рівні безпеки $sl_{Di} \in MS$. Тоді множина рівнів безпеки SL_{DB} елементів у СБД визначається як $SL_{DB} = \{SL_0, SL_M\}$, де SL_M – множина рівнів безпеки триплетів у *M*, SL_0 – множина рівнів безпеки всіх елементів онтології *O*.

Як було відзначено вище, в онтології *O* містяться множини класів *3* (понять) і властивостей *P;* між класами існують відносини підкласів *(sub-ClassOf),* а властивостями – відношення підвластивість *(subPropertyOf).* У зв'язку із цим повинне виконуватися наступне узгодження рівнів безпеки елементів онтології:

– $sl_{Csub} \geq sl_{Csup}$, де sl_{Csub} – рівень безпеки підкласу, *slcsub* – рівень безпеки суперкласу;

– $sl_{ix} \geq sl_{Cy}$, де sl_{ix} – рівень безпеки індивіда i_x ($i_x \in$ I, де *1*= $\{i_1 \ldots, i_к\}$ – множина індивідів класу c_y онтології), sl_{Cy} – рівень безпеки класів $\boldsymbol{c_y}$, яким індивід *ix* належить;

– $sl_{Psub} \geq sl_{Psup}$, де sl_{Psub} – рівень безпеки підвластивість p_{sub}, p_{sup} – рівень безпеки інших властивостей *psup*, яким *psub* належить.

Множина рівнів безпеки всіх елементів онтології SL_0 може бути визначене як $SL_0 = \{SL_C, SL_P\}$, де $SL_C = \{sl_{C1}, ..., sl_{Cm}\}$ – множина усіх рівнів безпеки класів в

онтологіях СБД, а $SL_P = \{sl_{p1} ..., sl_{pn}\}$ – множина рівнів безпеки властивостей онтологій СБД.

Для контролю можливості виконання логічних правил $r_i \in R = \{r1, ..., r_k\}$ кожному логічному правилу повинен задаватися рівень безпеки sl_r. Тоді множину рівнів безпеки результатів логічних правил можна позначити як $SL_R = \{slr_1, ..., slrk\}$, де *slri* – рівень безпеки логічного правила r_i. Користувач може використовувати логічні правила r_i для отримання результатів логічних висновків тільки в тому випадку, якщо $sl_U \geq sl_{ri.}$

В DB_S зберігається множина триплетів $T_{DB} = \{T_O, T_M\}$, де T_O – множина триплетів онтологій, T_M – множина триплетів семантичних метаданих.

Безпека СБД оцінюється захищеністю кожного триплету. Отже, семантична БД DB_S є безпечною тільки у випадку, коли всі триплети $t_{dbi} \in T_{DB}$ є захищеними.

Кожному триплету t_{Dbi} може бути заданий початковий рівень безпеки $sl_{DBi} \in MS$.

Позначивши пари триплета t_{Dbi} і його рівень безпеки sl_{DBi} як $sc_{DBi} = (t_{Dbi}, sl_{Dbi})$. Тоді безпека СБД характеризується множиною $SC_{DB} = \{T_{DB}, SL_{DB}\}$, яке називається покриттям безпеки.

Під покриттям безпеки СБД розуміється множина $SC_{DB} = \{T_{DB}, SL_{DB}\}$, де T_{DB} – множина усіх триплетів у СБД, SL_{DB} – множина усіх рівнів безпеки SL_{DBi}, що відповідають $t_{DBi} \in T_{DB}$. Відповідність (sl_{DBi}, t_{DBi}) повинне бути однозначним.

Кожна пара $sc_{DBi} = (t_{DBI}, sl_{DBI})$, що полягає із триплету і його рівня безпеки, називається парою безпеки триплету.

За допомогою логічних правил R користувач U може отримати результати логічних висновків R_L. Елементам в онтології O и триплетам t у метаданих M можуть бути задані рівні безпеки $sl_i \in MS$. Тоді отримувані результати логічних висновків також мають рівні безпеки $sl_{RL} \in MS$. Безпека кожного результату логічних висновків оцінюється $sc_l = (rl, sl_{RL})$, де $r_L \in R_L$ – результат логічних висновків, sl_{RL} – рівень безпеки r_L.

Множина усіх пар $sc_l = (r_L, sl_{RL})$, отриманих у СБД, називається покриттям безпеки результатів логічних висновків, яке позначається як $SCL = \{R_L, SL_{RL}\}$.

Для вибірки інформації із семантичних БД використовуються запити, описані мовою SPARQL [35]. Загальна структура SPARQL-запиту q до семантичних БД показана на рисунку 3.1.

```
PREFIX # префіксні оголошення
FROM... # джерела запиту
SELECT # пункт результату
WHERE {...} # шаблон запиту
ORDER BY # модифікатори запиту
```

Рисунок 3.1 – Загальна структура SPARQL-запиту

У загальній структурі SPARQL-запиту шаблон «WHERE» є множиною моделей RDF-триплетів і позначається як $SP = \{pt_1, ..., pt_n\}$. Шаблон SP визначає, що запитувати із семантичної БД.

У шаблоні p_{ti} – є моделлю триплету, яка складається з набору трійок: суб'єкт (s), предикат (p), об'єкт (o), де кожний компонент є: константою, такий, як $s \in C$, $p \in P$, і $o \in C \cup L$ або змінної, представленої у вигляді символів «$» або «?».

У таблиці 3.2 показана множина 8 можливих видів моделей RDF-триплету pt_i, яке позначається як $Pt = \{pt_1, ..., pt_8\}$.

Таблиця 3.2. – Усі можливі види моделей RDF-триплету

№	Select	Опис
1	[s, p, o]	Всі елементи триплету є константами
2	[s, ?x, o]	Суб'єкт і об'єкт є константами, а предикат є змінної
3	[s, p, ?x]	Суб'єкт і предикат є константами, а об'єкт є змінної
4	[?x, p, o]	Об'єкт і предикат є константами, а суб'єкт є змінної
5	[s, ?x, ?y]	Об'єкт і предикат є змінними, а суб'єкт є константою
6	[?x, p, ?y]	Суб'єкт і об'єкт є змінними, а предикат є константою
7	[?x, ?y, o]	Суб'єкт і предикат є змінними, а об'єкт є константою
8	[?x, ?y, ?z]	Всі елементи є змінними.

Якщо шаблон *SP* запиту *q* збігається з якою-небудь частиною (тілом або головою) логічного правила, то відповідь на даний запит буде визначатися в результаті логічного висновку.

Під логічним запитом розуміється запит *q* до семантичних БД, в якого шаблон «WHERE» збігається з тілом яких-небудь логічних правил $r_i \in R$. У результаті виконання цього запиту до СБД користувач може отримати результати логічних висновків.

Під прямим запитом розуміється запит *q* до семантичних БД, у якого шаблон «WHERE» не збігається з якими-небудь тілами яких-небудь логічних правил $r_i \in R$.

Наприклад, для логічного правила r_x ще має вигляд:

```
(?x є виконавцем ?y) (?z є продуктом ?y) → (?x є виробником ?z).
```

Якщо запит *q* має вигляд:

```
SELECT ?x ?в ?z WHERE {?x є виконавцем ?y. ?z є продуктом ?в},
```

то *q* є логічним запитом, тому що його шаблон збігається з тілом логічного правила.

Якщо *q* має інший вигляд:

```
SELECT ?x ?y ?z WHERE {?x є виконавцем ?y. ?z описує ?y},
```

або

```
SELECT ?x ?y ?z WHERE {?x є виконавцем ?y},
```

то *q* є прямим запитом, тому що його шаблон не збігається з жодним тілом логічного правила $\boldsymbol{r_x}$.

3.3 Розробка алгоритму підтримки безпеки роботи із семантичними БД

Зареєстровані користувачі *U*, що мають рівень доступу sl_U і права доступу U_p, можуть відправляти SPARQL-запити *q* до семантичної БД DB_S для перегляду,

отримання, додавання або зміни їх даних *D* і для виконання логічних правил *R* для отримання результатів логічних висновків r_L.

Семантична БД має вважатися безпечною, якщо задовольняє наступним умовам:

– користувач *U* може мати право доступу на перегляд даних *D*, якщо $sl_U \geq sl_D$;

– користувач *U* має права на зміну, видалення й додавання даних D якщо $sl_U \geq sl_D$ *i* U_P = {перегляд, зміна, видалення, додавання};

– користувач U може виконувати логічні правила **r**, якщо $sl_U \geq sl_R$;

– користувач може отримати результати логічних висновків r_l, якщо $sl_U \geq sl_{rL}$.

На рисунку 3.1 показаний загальний процес забезпечення безпеки СБД. Процес забезпечення безпеки СБД виконується в наступний спосіб – якщо запит користувачів до СБД є прямим запитом, то модуль «керування доступом» виконує наступні дії:

– перевіряє рівні доступу користувачів;

– визначає рівні безпеки отриманих відповідей на запит;

– дає результати користувачам відповідно до їхніх рівнів доступу.

Якщо запит є логічним запитом, то модуль «підсистема виконання логічних висновків» виконує наступні дії:

– перевіряє рівні доступу користувачів;

– визначає можливість виконання логічних правил;

– виконує логічні висновки;

– виявить порушення результатів логічних висновків;

– контролює результати логічних висновків;

– дає результати користувачам відповідно до їхніх рівнів доступу.

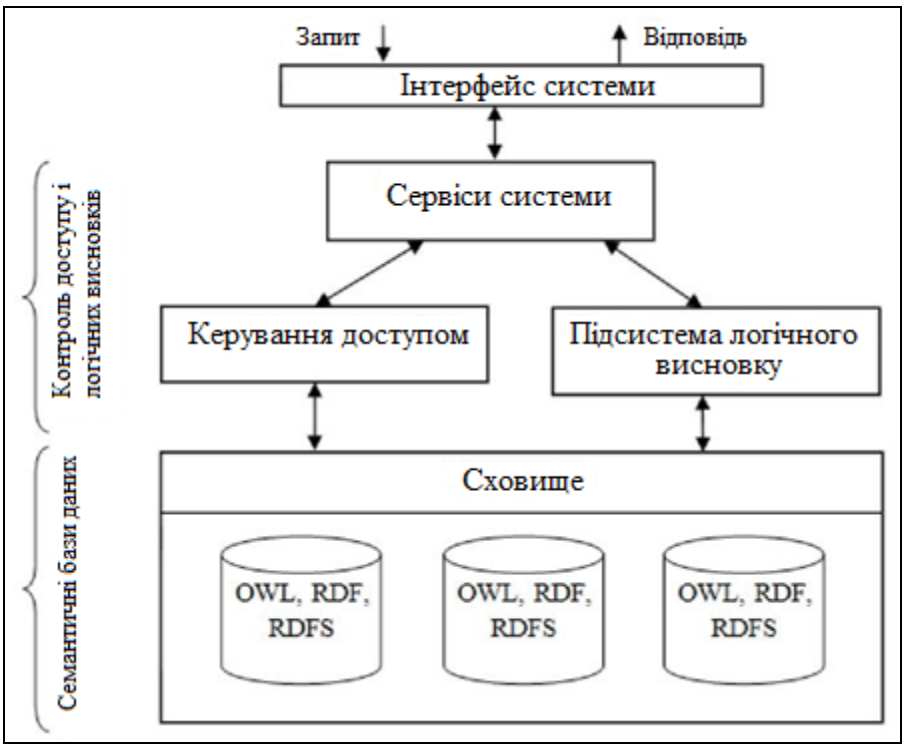


Рисунок 3.1 – Алгоритм забезпечення безпеки роботи із семантичними БД

3.4 Архітектура пропонованої системи підтримки безпеки

Для забезпечення безпеки СБД потрібно побудувати систему забезпечення безпеки роботи із семантичними БД (позначену як SS), під якою розуміється система, що володіє двома можливостями: контролем доступу користувачів до окремих елементів СБД і контролем результатів логічних висновків.

Пропонована архітектура системи SS розділена на 6 рівнів, відповідних до різних етапів обробки запитів користувачів (див. рис. 3.2).

Інтерфейс системи забезпечує взаємодію між користувачами й системою, за допомогою його користувачі можуть відправити запити до системи й отримати відповіді на них.

Рівень представлення даних дає відповіді користувачам. Відповіді можуть бути оформлені з використанням різних форматів даних, наприклад таких, як RDF/XML, Turtle або N3.

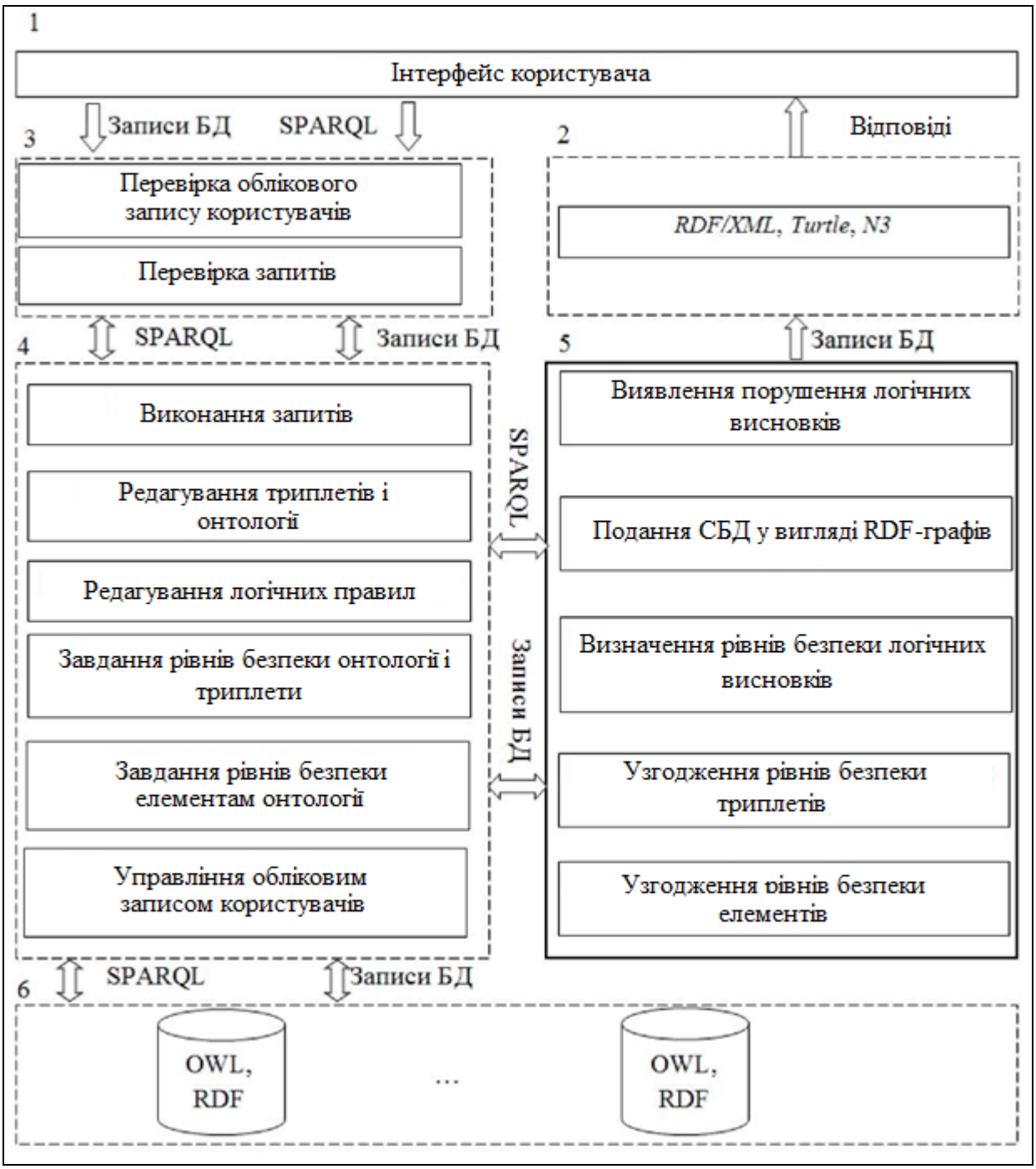


Рисунок 3.2 – Розроблена структура системи забезпечення безпеки роботи із семантичними БД

Рівень підготовки даних здійснює перевірку інформації користувачів і коректність заданих запитів.

Рівень сервісів системи – основні функціональності системи, які користувачі можуть виконувати, такі, як перегляд даних, модифікація триплетів, модифікація онтології, задача рівнів безпеки триплетам, задача рівнів безпеки

елементам онтології й керування обліковим записом користувачів і т.ін.

Рівень 5 – рівень забезпечення безпеки – основна частина системи підтримки безпеки семантичних БД**.** На рисунку 3.2 показані типові функції даної частини, які розробляються й досліджуються в даній роботі. Вони реалізуються у вигляді наступного набору модулів:

– модуль узгодження рівнів безпеки елементів онтологій і індивідів метаданих;

–модуль визначення рівнів безпеки триплетів – здійснює процес розрахунку рівнів безпеки всіх можливих триплетів у СБД;

–модуль визначення рівнів безпеки результатів логічних висновків – визначає всі рівні безпеки триплетів, отриманих на основі відомих даних за допомогою використання логічних правил;

–модуль представлення семантичної БД у вигляді RDF-графів – визначає всі RDF-графи в СБД;

–модуль виявлення порушень результатів логічних висновків – визначає всі несанкціоновані отримані триплети при виконанні логічних правил.

Рівень 6 – Сховище баз даних використовується для зберігання RDF-метаданих, онтології предметних областей і логічних правил.

На основі запропонованої архітектури системи забезпечення безпеки роботи із семантичними БД підтримується виконання процесів, показаних на рисунку 3.3.

Усі процеси роботи користувачів розділені на групи

Група I – користувачі: користувачі мають свої облікові записи й можуть виконувати функціональності, такі, як модифікація триплетів, задача рівнів безпеки триплетам і перегляд даних.

Залежно від обраної функціональності система формує відповідний запит, який відправляється оброблювачу. Оброблювач обробляє дані в СБД і викликає компонентів «забезпечення безпеки» для підтримки безпеки даних при виконанні відповідної функціональності. Результати виконання роботи видаються користувачам.

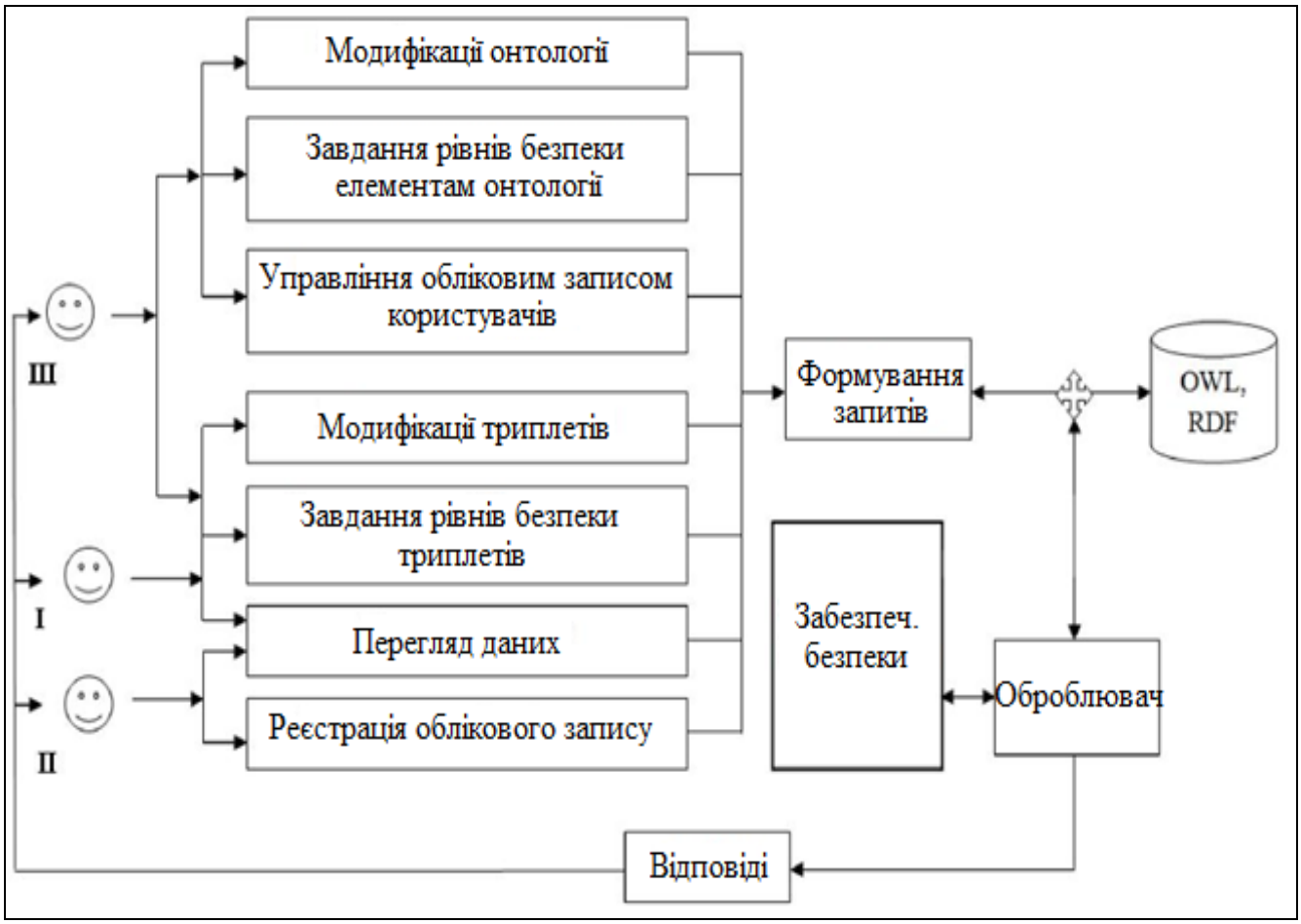


Рисунок 3.3 – Основні процеси роботи користувачів системи

Залежно від обраної функціональності система формує відповідний запит, який відправляється оброблювачу. Оброблювач обробляє дані в СБД і викликає компонентів «забезпечення безпеки» для підтримки безпеки даних при виконанні відповідної функціональності. Результати виконання роботи видаються користувачам.

Група II – гості: користувачі низького статусу можуть виконувати функціональності перегляду даних і реєстрацію до системи.

Група III – адміністратори: користувачі високого статусу мають можливість виконувати все функціональності системи, таких як перегляд, модифікація, задача рівнів безпеки даним (триплетам і елементам онтології, логічним правилам), керування обліковим записом користувачів.

Пропонована система забезпечує безпеку семантичної БД у результаті виконання наступних дій (див. рис. 3.4):

Рисунок 3.4 – Процес забезпечення безпеки СБД

При початковому запуску система виконує наступні дії:

– визначає рівні безпеки SL_{DB} усіх елементів онтології O и метаданих M с допомогою модуля «узгодження рівнів безпеки елементів онтології»;

– визначає рівні безпеки SC_{DB} усіх триплетів у СБД за допомогою модуля «узгодження рівнів безпеки триплетів»;

– визначає рівні безпеки SC_L можливих результатів логічних висновків СБД за допомогою модуля «визначення рівнів безпеки результатів логічних висновків».

Покриття безпеки триплетів *SCDB* і покриття безпеки можливих результатів логічних висновків $SCDB$ зберігаються в СБД.

При кожному вході користувача U у систему за допомогою модуля «перевірка обліковому запису користувачів» виконується перевірка наявності його обліковому запису, рівня доступу sl_U і прав доступу U_p, інформація про яких зберігається в СБД. Якщо перевірка закінчується успішно, то користувач може відправляти запити q до системи для виконання різних операцій над даними відповідно до його рівнів і правами доступу.

При відправленні користувачем запиту q система виконує його перевірку за допомогою модуля «перевірка запиту».

Якщо запит q складено граматично неправильно, то q не виконується, і користувач повинен сформувати інші запити до системи.

Якщо q є логічним запитом (q збігається з логічним правилом **r)** і якщо $sl_U \geq sl_R$ (рівень доступу користувачів не менше рівня безпеки логічного правила), то запит q може виконуватися для отримання результатів логічних висновків, інакше, якщо $sl_U < sl_R$, то запит q не виконується.

Якщо q є прямим запитом, то він може виконуватися.

Система дає користувачеві результати відповідно до політики безпеки:

– при виконанні прямого запиту q система дає користувачеві U відповіді A, у яких рівні безпеки sl_A не більше рівня доступу користувачів sl_U ($sl_A \leq sl_U$);

– при виконанні логічного запиту система виконує виявлення порушень результатів логічних висновків за допомогою модуля «виявлення порушень результатів логічних висновків».

У результаті цього система дає користувачеві дозволені відповіді.

В онтології O клас c_x може бути підкласом класу c_y ($c_x \in c_y$) або суперкласом класу c_z ($c_z \in c_x$). Якщо користувачі U мають доступ до підкласу c_x, то вони можуть мати доступ і до його суперкласу c_y, і навпаки, якщо користувачі мають доступ до суперкласу cy, то вони можуть і не мати доступ до підкласу cx. На підставі цього забезпечується, що рівень безпеки sl_{cyx} підкласу c_x не менше рівня безпеки sl_{Cy} його суперкласу c_y ($sl_{Cx} \geq sl_{Cy}$).

Крім цього, якщо між класами c_x і c_y є такі зв'язки, як *owl:sameAs* і *owl:equivalentClass*, то користувачі можуть отримати доступ до екземплярів класу

c_x через екземпляри класу *cy* й навпаки. У зв'язку із цим, такі класи повинні мати однаковий рівень безпеки, тобто, $sl_{Cx} = sl_{Cy}$.

Аналогічно, для властивостей $p_x p_y$ онтології *O* повинні виконуватися наступні умови для їхніх рівнів безпеки.

Якщо властивість p_x є підвластивості властивості p_y ***(p_x*** включається *в* p_y*)*, то $sl_{px} \geq sl_{py}$, де sl_{px} – рівень безпеки властивості, sl_{py} – рівень безпеки його супервластивості p_y.

Якщо між властивостями p_x і p_y існують такі відносини, як *owl:inverseOf, owl:equivalentProperty* або *owuinversefunctionalProperty,* то користувач може визначити значення властивості p_x через значення властивості p_y й навпаки. У зв'язку із цим, такі властивості повинні мати однаковий рівень безпеки, тобто $sl_{Cx} = sl_{Cy}$. Як приклад таких властивостей можна привести пари: *«isparentOf»* – *«ischildenOf», «isteacherOf»* – *«isstudentOf»* і т.д.

У СБД кожний клас $c_y \in C$ може містити в собі множину індивідів $i_y = \{i_1, ..., i_h\}$. Для отримання доступу до якого-небудь індивіда $i_j \in c_y$ користувач повинен має права доступу до класу c_y, а це позначає, що рівень безпеки індивідів sl_{Ij} не повинен бути менше рівня безпеки класу sl_{Iy}, до якого він ставиться ($sl_{Ij} > sl_{Cy}$).

Спочатку в СБД елементам онтологій (клас і властивість) і індивідам можуть бути задані будь-які початкові рівні безпеки, у зв'язку із цим виникає така проблема:

– рівень безпеки підкласу менше рівня безпеки суперкласу;

– рівень безпеки підвластивості менше рівня безпеки супервластивості;

– у класів, що мають зв'язку *owl.sameAs, owl:equivalentClass,* різні рівні безпеки;

– у властивостей, що мають зв'язки *owl:inverseOf, owl:equivalentProperty* або *owl:unverseFunctionalProperty,* різні рівні безпеки;

– рівень безпеки індивіда не менше рівня безпеки класу, у який він включається.

Відповідно до політики безпеки СБД в DB_S даним *D* задаються рівні безпеки $sl_D \in MS$. Кожному користувачеві *U* задаються права доступу *Up*, значення яких

приймаються з множини прав {читання (*read*), запис (*write*), виконання (*execute*), зміна (*modify*)} і рівень доступу $sl_U \in MS$. Користувач має доступ до даних тоді й тільки тоді, коли $sl_U \geq sl_D$, таким чином, для контролю доступу користувачів до даних у СБД необхідно визначити рівні безпеки всіх їхніх елементів.

У семантичних БД для забезпечення безпеки доступу до онтологій, класів і властивостям задаються початкові рівні безпеки $sl_i \in MS$. Множина рівнів безпеки всіх елементів онтології SL_O може бути визначене як $SL_O = \{SL_C, SL_P\}$, де $SL_C = \{slc_1, ..., sl_{Cm}\}$ – множина усіх рівнів безпеки класів в онтологіях СБД, а $SL_P = \{slp_1, ..., sl_{pn}\}$ – множина рівнів безпеки властивостей онтологій СБД.

Початкові рівні безпеки елементів можуть бути неузгодженими, як показано на рисунку 3.5. Рівні безпеки елементів у показаній частині є неузгодженими у зв'язку з наступними проблемами:

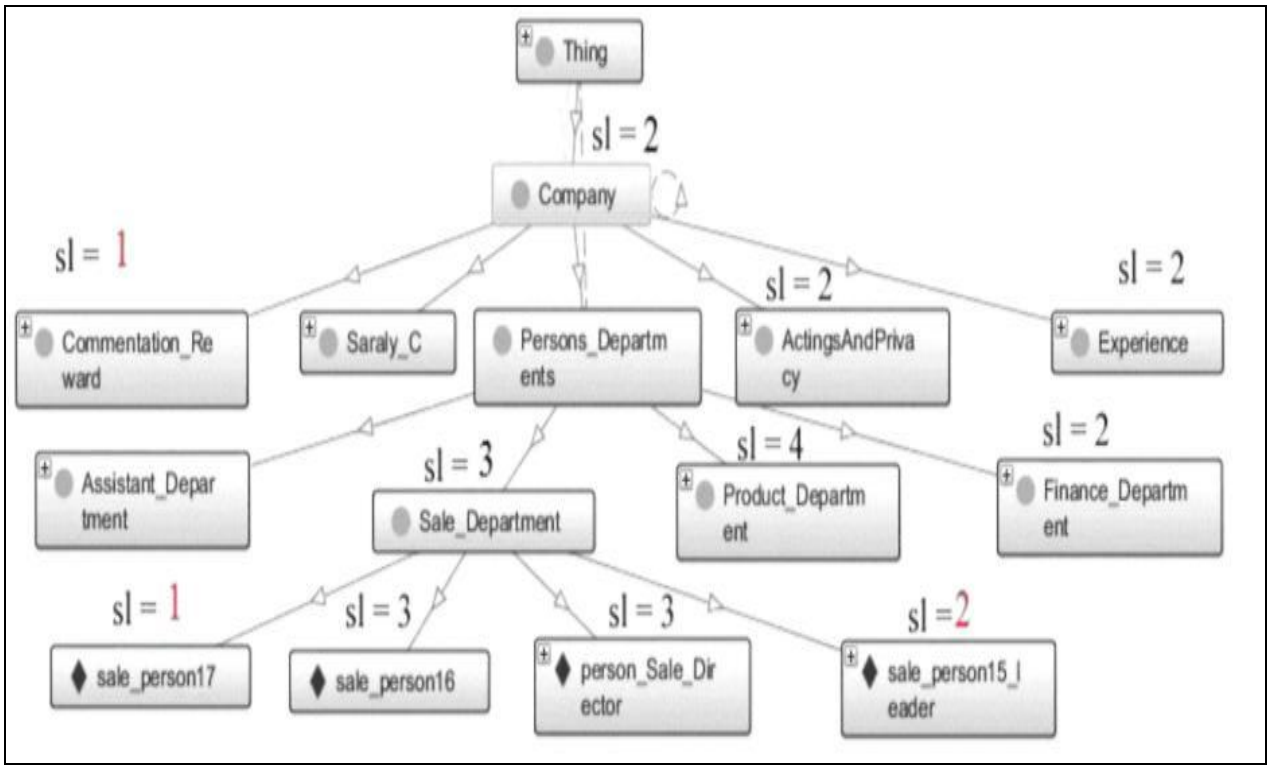


Рисунок 3.5 – Частина онтології з неузгодженими рівнями безпеки

Користувачі, що мають рівні доступу $sl_U = 1$, можуть мати доступ до підкласу «*Commentation_Reward*» класу «*Company*», тому що підклас «*Commentation_Reward*» має рівень безпеки $sl = 1$, $sl_U = sl$.

Але ці поля користувачі не можуть мати доступ до класу «*Company*», тому що в нього рівень без небезпеки $sl = 2 > sl_U = 1$, отже, користувачам не можна мати

доступ до яким-небудь його підкласам або індивідам. З обліком цього, захищеність онтології порушена.

Користувачі, що мають рівні доступу $sl_U = 2$, можуть мати доступ до індивідів *«salepersonl7»* і *«salepersonl5_leader»* класу *«sale_Department»*. Але з метою безпеки СБД будь-які користувачі, що мають рівні доступу $sl_U < 3$, не можуть мати доступ до яких-небудь індивідів у класі *«sale_Department»*, тому що його рівень безпеки $sl = 3 > sl_U$. З обліком цього, захищеність даної онтології порушена.

Для контролю доступу користувачів до елементів онтології необхідно погодити рівні безпеки її елементів (класів, підкласів і індивідів).

4 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вибір засобів розробки пропонованої системи

Засоби побудови системи підтримки безпеки роботи із семантичними БД відіграють важливу роль для оцінки ефективності її функціонування. Для опису онтології використовується мова OWL, для створення онтології пропонується використовувати редактор Protégé, для зберігання семантичних даних використовується система Virtuoso Universal, для опису логічних правил використовується мова SWRL.

Для розробки семантичних додатків використовується фреймворк Jena. Jena – це Java API (інтерфейс програмування додатків Java) для розробки додатків Semantic Web, що має наступні переваги:

– підтримка прикладного програмного інтерфейсу для роботи з RDF (RDF API);

– підтримка прикладного програмного інтерфейсу для роботи з OWL (OWL API), який може бути використаний і в якості RDFS API;

– можливість читати й записувати RDF у різних форматах: RDF/XML, N3 і N-triples;

– можливість зберігати RDF-моделі постійно в пам'яті;

– підтримка роботи з SPARQL-запитами;

– підтримка роботи з логічними правилами й реалізація їх виконання для отримання нової інформації із БД.

Для контролю доступу користувачів до триплетів і елементам семантичних БД розроблений ряд алгоритмів: узгодження рівнів безпеки елементів онтологій і індивідів метаданих; визначення покриття безпеки семантичних БД; керування відповідями при виконанні запитів до СБД. На їхній основі розроблена загальна структура роботи програми контролю доступу користувачів до семантичних БД, яка показано на рисунку 4.1.

Загальний опис програми наведено в таблиці 4.1

Таблиця 4.1 – Характеристики програми підтримки безпеки

Програма:	Програма підтримки безпеки роботи із семантичними базами знань.
Анотація:	Програма призначена для забезпечення безпеки роботи із семантичними базами знань, що включають онтології (OWL) і екземпляри (RDF-дані). Вона може бути використана в інформаційних системах організацій. Програма забезпечує виконання наступних функцій: – перевірку облікового запису користувачів; – читання/запис семантичних метаданих у наступних форматах: RDF/XML, RDF/N3, RDF/Turtle, RDF/N-Triples; – задача рівнів доступу користувачів і рівнів безпеки триплетів онтологій і метаданих у СБД; – перевірку відповідності рівнів доступу користувачів до елементам онтологій; – виконання SPARQL-запитів з урахуванням прав доступу користувачів; – редагування (видалення, додавання й модифікацію) триплетів семантичних даних залежно від прав доступу користувачів.
Мова:	Java
ОС:	MS Windows 7/10
Обсяг програми:	300 Мб.

Користувацький інтерфейс побудований для взаємодії між користувачами й системою, за допомогою його користувач може вибирати різні функціональності системи для роботи зі СБД і отримати відповіді на них.

Рівень представлення даних дає відповіді користувачам. Відповіді можуть бути оформлені з використанням різних форматів даних, наприклад таких, як *RDF/XML*, *Turtle* або *N3*.

Рівень підготовки даних здійснює перевірку інформації користувачів і коректність заданих запитів. Він складається з наступних модулів:

– модуль перевірки облікової записів користувачів, який використовується для визначення їх рівнів і права доступу;

– модуль перевірки запиту, який визначає коректність створеного запиту; кожний запит повинен бути правильно складений відповідно до синтаксису

використовуваного мови запитів.

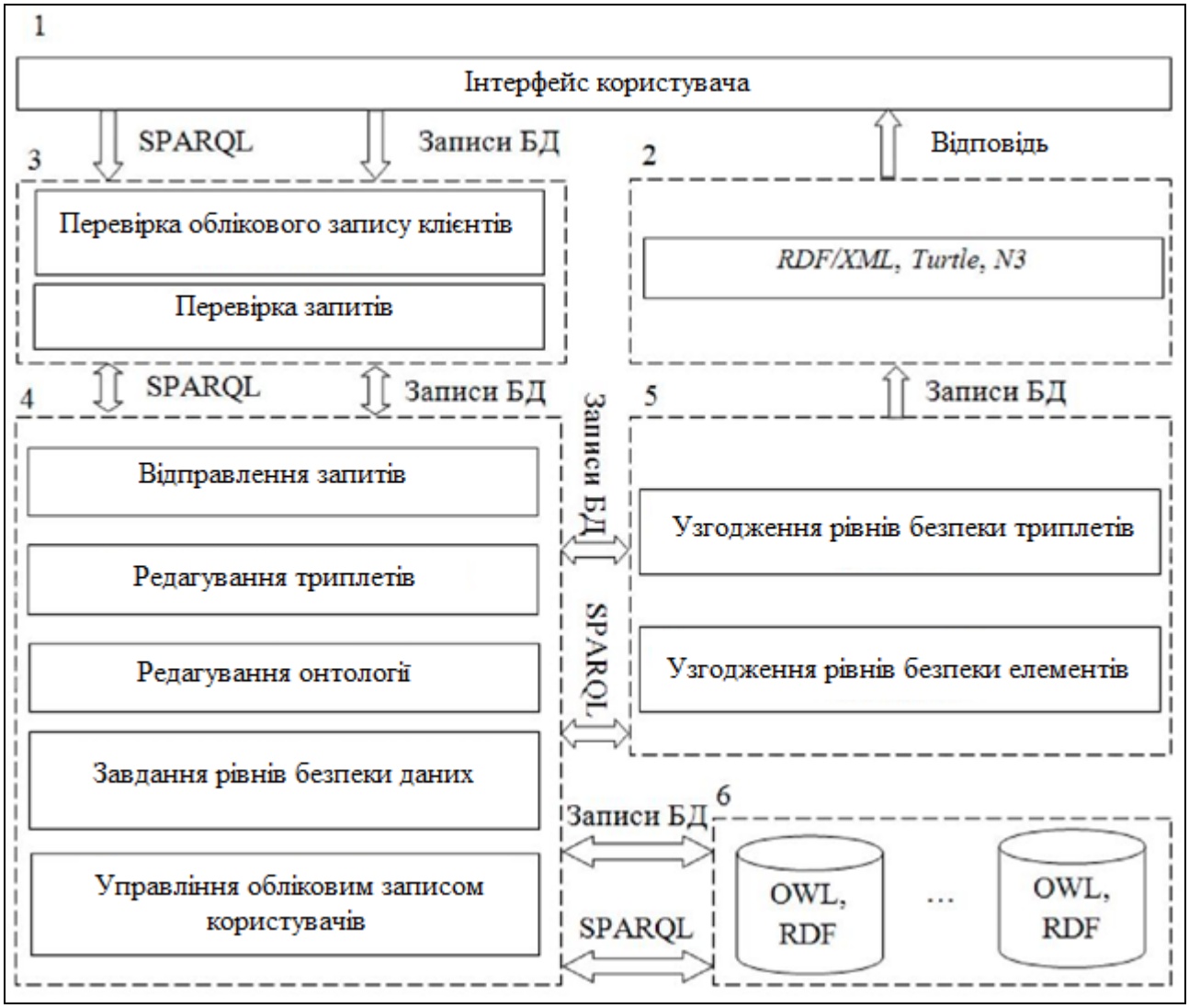


Рисунок 4.1 – Загальна структура програмного забезпечення контролю доступу користувачів до СБД

Рівень сервісів системи – основні функціональності системи, які користувачі можуть виконувати, такі, як редагування триплетів, редагування онтології, задача рівнів безпеки даних, відправлення запитів, керування обліковим записом користувачів.

Рівень контролю доступу користувачів до семантичних БД є основним компонентом програми, у якому містяться наступні модулі:

– модуль визначення рівнів безпеки елементів онтології й індивідів метаданих;

– модуль визначення рівнів безпеки триплетів СБД (визначення погоджених рівнів безпеки триплетів СБД).

Семантична БД використовується для зберігання семантичних даних, у якій зберігаються онтології й метадані. В якості RDF-сховища обрана система Virtuoso Universal Server.

Процес контролю доступу користувачів до елементів БД виконується в такий спосіб:

При кожному вході користувача в систему за допомогою модуля «перевірка обліковому запису користувачів» виконується перевірка наявності його облікового запису, рівня доступу й прав доступу, інформація про яких зберігається в СБД.

При відправленні користувачем запитів система виконує їхню перевірку за допомогою модуля «перевірка запиту».

Система визначає рівні безпеки всіх елементів онтологій і триплетів у СБД.

Система дає користувачам відповіді на запит відповідно до його рівнів безпеки.

Дана програма гарантує, що користувачі виконають операції над даними семантичних БД і отримати результати відповідно до їхніх рівнів і правами доступу. На рисунку 4.3 показаний графічний інтерфейс програми підтримки безпеки роботи із семантичними базами знань.

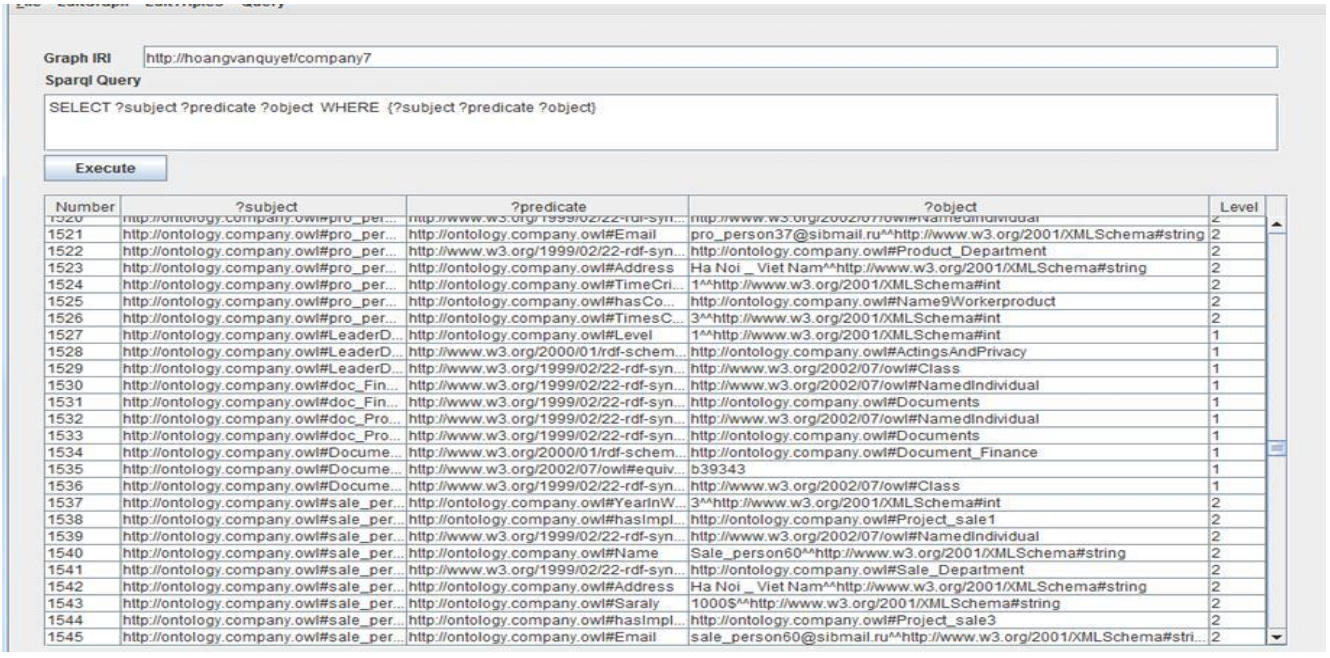


Рисунок 4.3 – Виконання запиту користувачів до СБД

Дана програма дозволяє контролювати доступ користувачів до окремих елементів даних. Користувачі можуть виконувати різні операції над даним у відповідності їх правам і рівням доступу.

Наприклад, користувач, що має рівень доступу рівний 2, отримує тільки триплети, у яких рівні безпеки не більше 2.

Користувач, що має права доступу на додавання даних може додавати дані в СБД (див. рис. 4.4).

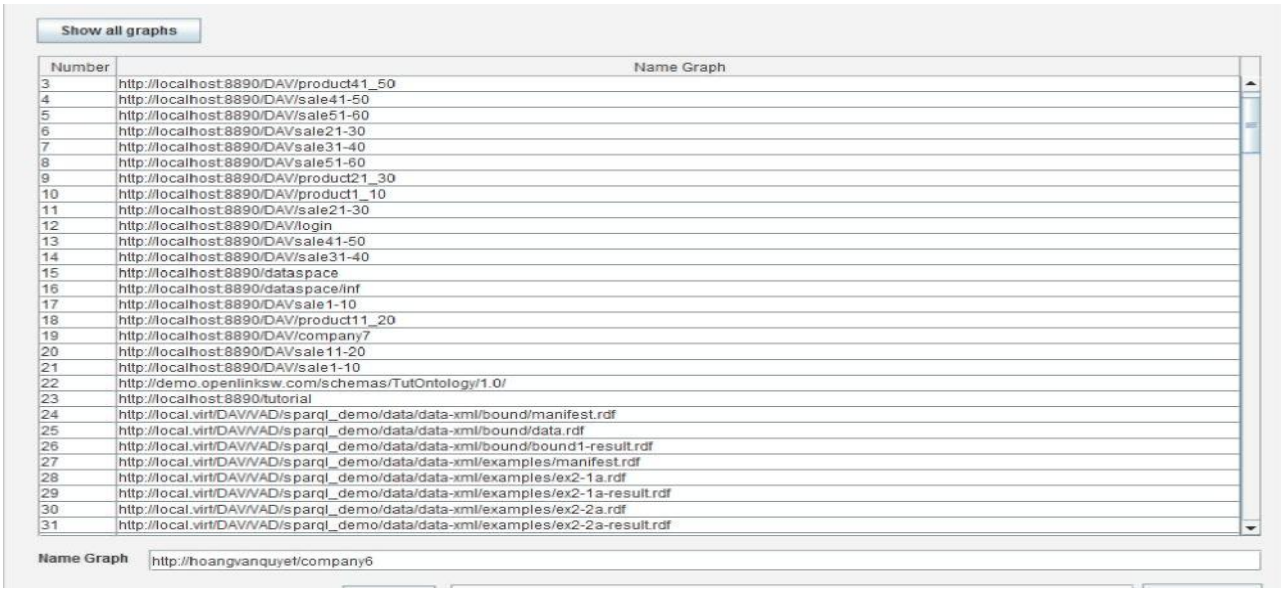


Рисунок 4.4 – Додавання RDF-даних у СБД

На рисунку 4.5 показана операція зміни RDF-триплету в СБД.

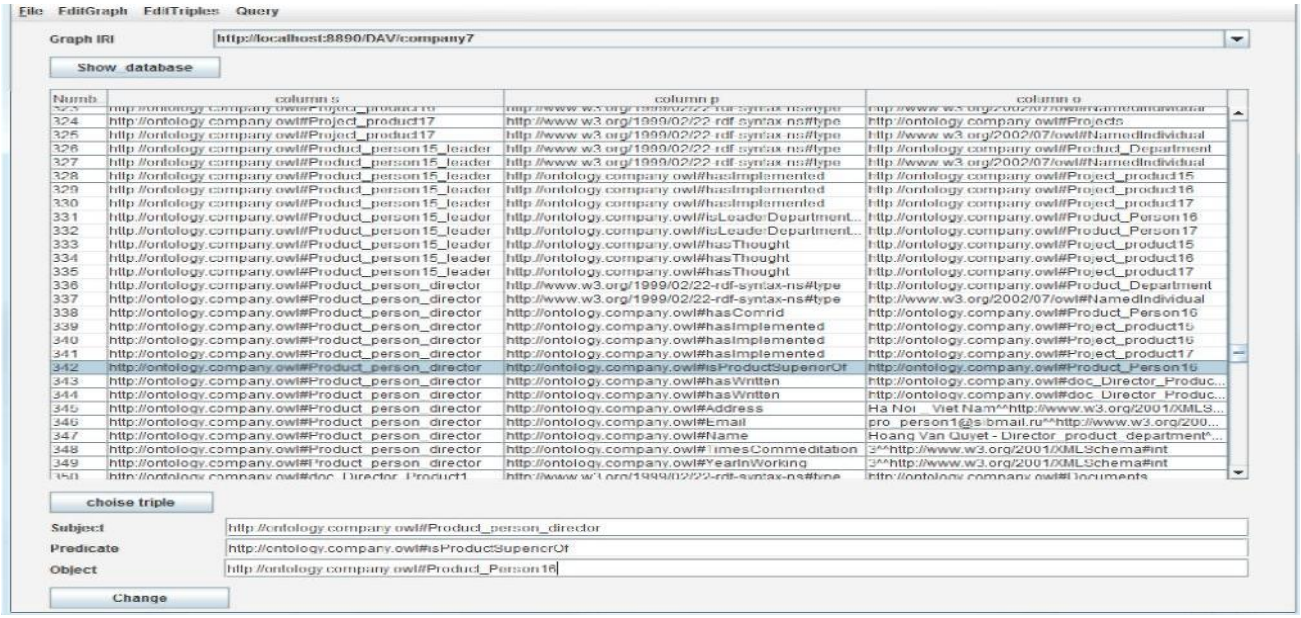


Рисунок 4.5 – Редагування RDF-триплету в СБД

4.2 Програмна реалізація алгоритму контролю логічних висновків

Для реалізації програмного забезпечення контролю результатів логічних висновків СБД використовуються методи й алгоритми, такі, як представлення СБД у вигляді RDF-графа, визначення можливості отримання результатів логічних висновків між двома вершинами; виявлення порушень результатів логічних висновків у семантичній БД; керування відповідями при виконанні запитів до СБД.

На основі розроблених алгоритмів запропонована загальна структура роботи програми контролю результатів логічних висновків у семантичних БД, як показано на рисунку 4.5.

Дана програма складається з наступних рівнів:

Користувацький інтерфейс побудований для взаємодії між користувачами й системою, за допомогою його користувач може вибирати різні функціональності системи для роботи зі СБД і отримати відповіді на них.

Рівень представлення даних дає відповіді користувачам, вони можуть бути оформлені з використанням різних форматів даних, наприклад таких, як *RDF/XML, Turtle* або *N3.*

Рівень підготовки даних здійснює перевірку інформації користувачів і коректність заданих запитів.

– модуль перевірки облікової записів користувачів, який використовується для визначення їх рівнів і права доступу;

– модуль перевірки запиту, який визначає коректність створеного запиту; кожний запит повинен бути правильно складений відповідно до синтаксису використовуваного мови запитів, наприклад SPARQL;

Загальний опис програми контролю наведено в таблиці 4.2.

Таблиця 4.2 – Характеристики програми ContrLSD

Програма:	Програма контролю логічних висновків у семантичних БД
Анотація:	Програма призначена для контролю можливості отримання висновків користувачами семантичних БД недозволених їм результатів логічних висновків. Вона може бути використана в інформаційних системах організацій. Програма забезпечує виконання наступних функцій: – перевірку облікового запису користувачів; – завантаження логічних правил, що зберігаються в СБД; – визначення рівнів доступу користувачів і рівнів без небезпеки триплетів у СБД; – виконання SPARQL-запитів до СБД; – визначення можливості виконання логічних правил, що дозволяють отримувати результати, що перевищують права доступу користувачів.
Мова:	Java
ОС:	MS Windows 10
Обсяг програми:	350 Мб.

Рівень сервісів системи – основні функціональності системи, які користувачі можуть виконувати, такі, як редагування триплетів і онтології, редагування логічних правил, задача рівнів безпеки елементом онтології й триплетам, задача рівнів безпеки елементам онтології, управління обліковим записом користувачів, виконання запитів.

Рисунок 4.5 – Загальна структура програмного забезпечення контролю результатів логічних висновків

Рівень контролю результатів логічних висновків семантичної БД є основним компонентом програми. На даному рівні дотримуються наступні основні модулі:

– модуль визначення рівнів безпеки елементів онтології й індивідів метаданих;

– модуль визначення рівнів безпеки триплетів СБД (визначення погоджених рівнів безпеки триплетів СБД);

– модуль визначення покриття безпеки результатів логічних висновків, розроблений на основі алгоритмів визначення рівнів безпеки результатів логічних висновків;

– модуль представлення RDF-графів, розроблений на основі алгоритму й алгоритму представлення семантичної БД у вигляді RDF-графів;

– модуль «виявлення порушень логічних висновків» визначає всі безпечні й розкриті елементи й триплети при виконанні конкретного запиту в СБД.

Семантична БД використовується для зберігання семантичних даних, у якій зберігаються онтології й метадані.

Опис реалізації програми контроля. Процес контролю запитів для виконання логічних висновків виконується в такий спосіб:

При кожному вході користувачів у систему, за допомогою модуля «перевірка обліковому запису користувачів» виконується перевірка наявності їх обліковому запису, рівня доступу й прав доступу, інформація про яких зберігається в СБД.

При відправленні користувачем запитів система виконує їхню перевірку за допомогою модуля «перевірка запиту».

Система визначає рівні безпеки результатів логічних висновків, отриманих при виконанні запиту.

Система виявляє порушення безпеки результатів логічних висновків користувачів.

Система видає користувачам результати відповідно до їхніх прав і рівнями доступу.

На рисунку 4.6 показаний графічний інтерфейс програми контролю логічних висновків СБД.

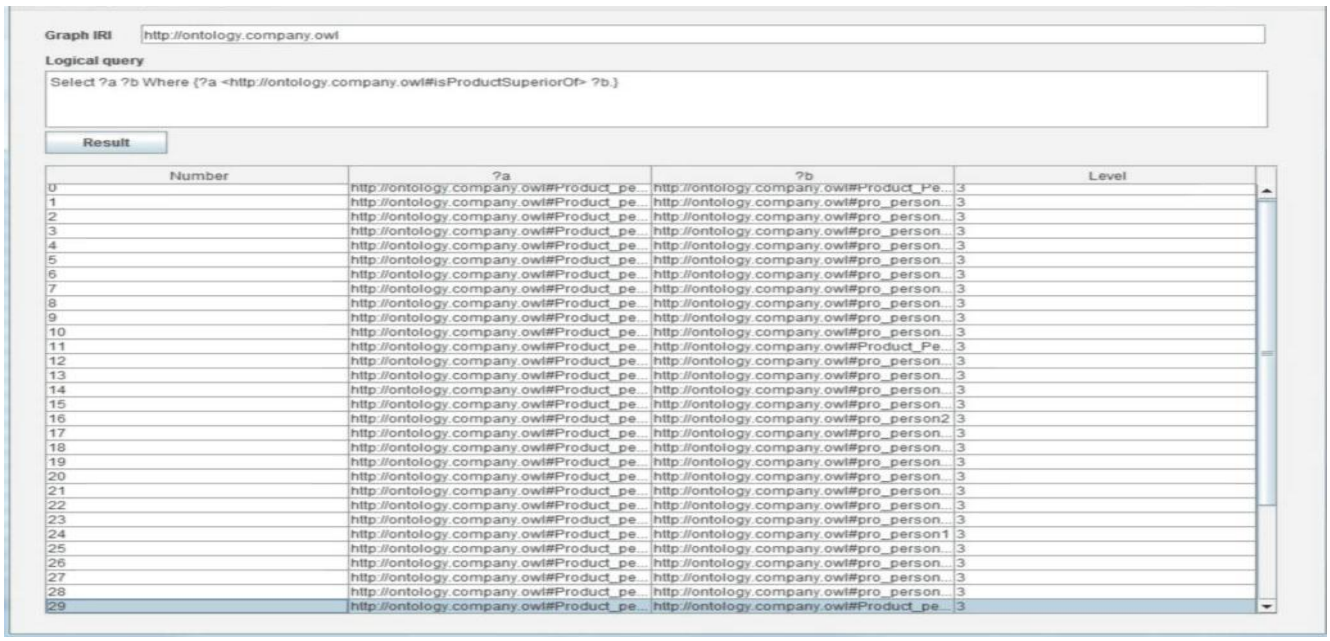


Рисунок 4.6 – Виконання запитів до СБД за допомогою програми контролю логічних висновків

Користувач, що має рівень доступу рівний 3, отримує тільки результати логічних висновків, у яких рівні безпеки не більше 3.

Розроблена система підтримує безпеку роботи із семантичними БД і гарантує, що користувачі зможуть отримати тільки ті результати логічних висновків, які відповідають їхнім рівням і правам доступу.

5 ОПИС МОЖЛИВОСТІ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У даному розділі наведені основні експерименти по дослідженню ефективності розроблених методів і алгоритмів, таких, як визначення погоджених рівнів безпеки елементів онтології; визначення рівнів безпеки всіх триплетів і результатів логічних висновків семантичних БД; виявлення порушень результатів логічних висновків; контроль результатів, отриманих при виконанні запитів до СБД.

Всі експерименти були проведені на персональному комп'ютері, що має наступну конфігурацію: процесор – AMD A8-4500M APU – 4 ядра – 1,90 Ghz; оперативна пам'ять – 4 GB.

У якості тестових даних використовувалася множина триплетів семантичної БД компанії ПП «РІІТ». Загальна кількість понять склала 162, кількість відносин – 137, кількість триплетів – 0.23 тисячи. Приклад частини онтологічної моделі наведено на рисунку 5.1.

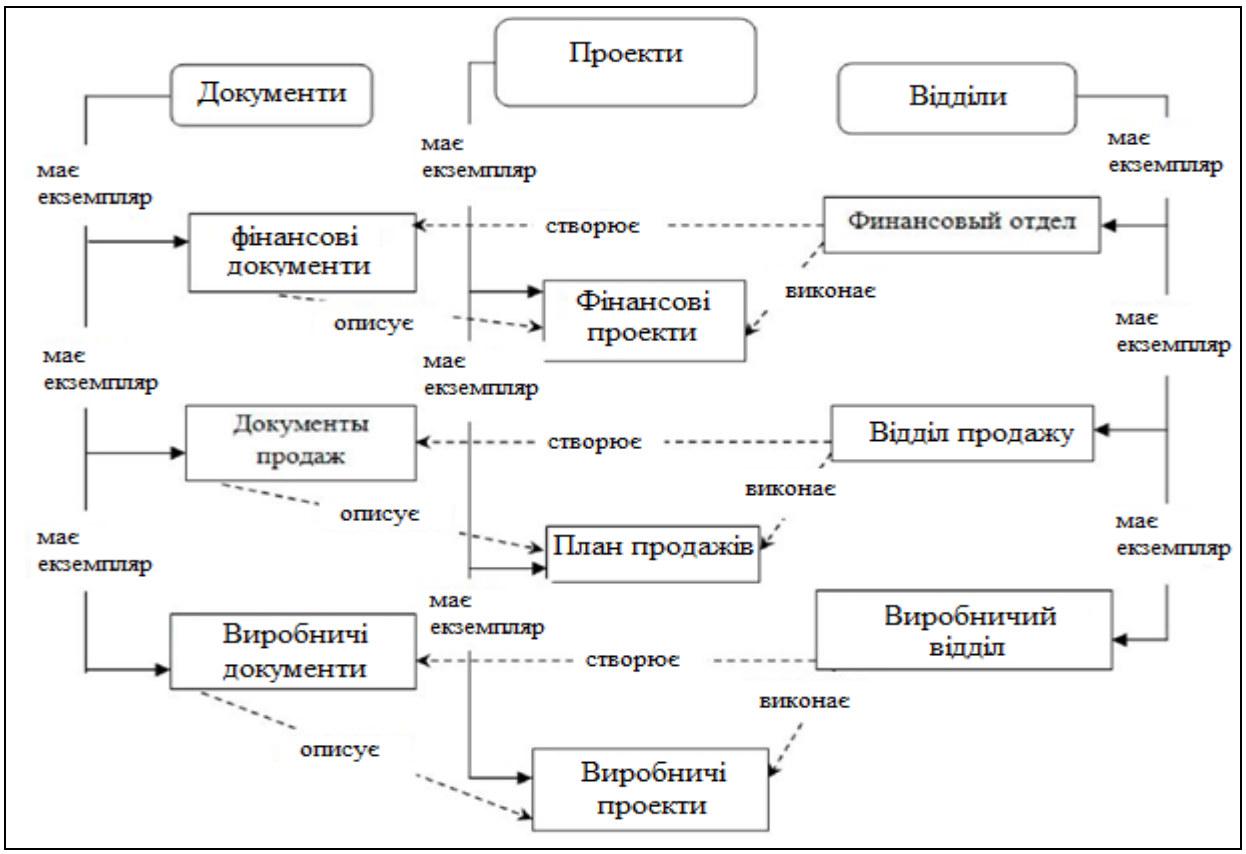
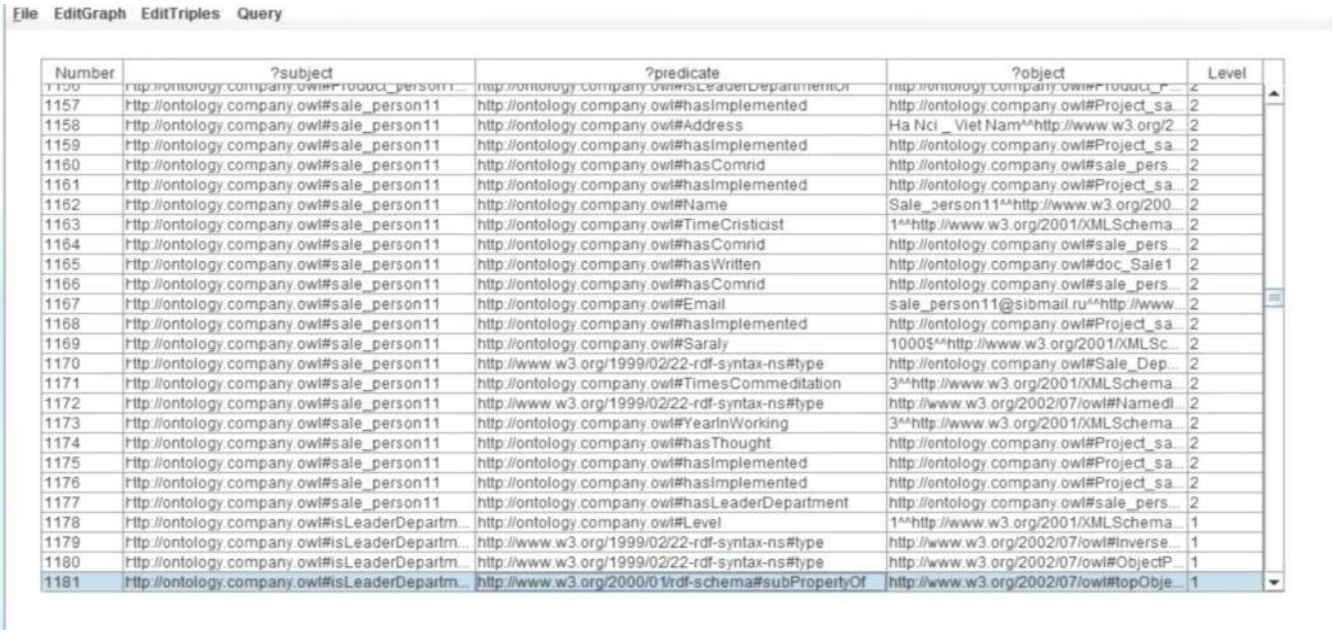


Рисунок 5.1 – Фрагмент використовуваної семантичної БД

Дані експерименти виконані за допомогою розробленої програми підтримки роботи із семантичними БД .

У першому експерименті в якості тестових даних усім елементам онтології і триплетам RDF-даних були задані рівні безпеки. У результаті виконання програми були визначені погоджені рівні безпеки всіх елементів онтологій і триплетів семантичної БД, як показано на рисунку 5.2.


File EditGraph EditTriples Query

Number	?subject	?predicate	?object	Level
1157	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1158	http://ontology.company.owl#sale_person11	http://ontology.company.owl#Address	Ha Noi _ Viet Nam^^http://www.w3.org/2...	2
1159	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1160	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasComrid	http://ontology.company.owl#sale_pers...	2
1161	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1162	http://ontology.company.owl#sale_person11	http://ontology.company.owl#Name	Sale_person11^^http://www.w3.org/200...	2
1163	http://ontology.company.owl#sale_person11	http://ontology.company.owl#TimeCristicist	1^^http://www.w3.org/2001/XMLSchema...	2
1164	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasComrid	http://ontology.company.owl#sale_pers...	2
1165	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasWritten	http://ontology.company.owl#doc_Sale1	2
1166	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasComrid	http://ontology.company.owl#sale_pers...	2
1167	http://ontology.company.owl#sale_person11	http://ontology.company.owl#Email	sale_person11@sibmail.ru^^http://www...	2
1168	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1169	http://ontology.company.owl#sale_person11	http://ontology.company.owl#Saraly	1000$^^http://www.w3.org/2001/XMLSc...	2
1170	http://ontology.company.owl#sale_person11	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ontology.company.owl#Sale_Dep...	2
1171	http://ontology.company.owl#sale_person11	http://ontology.company.owl#TimesCommeditation	3^^http://www.w3.org/2001/XMLSchema...	2
1172	http://ontology.company.owl#sale_person11	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Namedl...	2
1173	http://ontology.company.owl#sale_person11	http://ontology.company.owl#YearInWorking	3^^http://www.w3.org/2001/XMLSchema...	2
1174	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasThought	http://ontology.company.owl#Project_sa...	2
1175	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1176	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasImplemented	http://ontology.company.owl#Project_sa...	2
1177	http://ontology.company.owl#sale_person11	http://ontology.company.owl#hasLeaderDepartment	http://ontology.company.owl#sale_pers...	2
1178	http://ontology.company.owl#isLeaderDepartm...	http://ontology.company.owl#Level	1^^http://www.w3.org/2001/XMLSchema...	1
1179	http://ontology.company.owl#isLeaderDepartm...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Inverse...	1
1180	http://ontology.company.owl#isLeaderDepartm...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectP...	1
1181	http://ontology.company.owl#isLeaderDepartm...	http://www.w3.org/2000/01/rdf-schema#subPropertyOf	http://www.w3.org/2002/07/owl#topObje...	1

Рисунок 5.2 – Результат визначення покриття безпеки семантичних БД

Це показує, що алгоритм визначення покриття безпеки семантичних БД працює правильно.

Для дослідження ефективності даного алгоритму минулого проведено два дослідження:

– залежність часу визначення покриття безпеки від кількості триплетів і класів у семантичних БД;

– визначення необхідного обсягу оперативної пам'яті для зберігання колекції триплетів семантичних БД.

У таблиці 5.1 наведено результати визначення обсягів (мегабайт – МБ) простору для зберігання триплетів у СБД у трьох випадках:

Обсяг 1 простору семантичної БД для зберігання вихідних триплетів, у якій елементам не задані початкові рівні безпеки (обсяг 1);

Обсяг 2 простору семантичної БД для зберігання онтології й метаданих, у

якій елементам онтології задані початкові рівні безпеки, а рівні безпеки триплетів визначені за допомогою алгоритму визначення покриття безпеки БД (обсяг 2);

Обсяг 3 простору семантичної БД для зберігання онтологій і метаданих, у якій кожному елементу онтології й кожному триплету метаданих задані початкові рівні безпеки триплетів.

Таблиця 5.1 – Результати експерименту дослідження обсягів зберігання даних

Триплети	2	4	6	8	1	1	1	1
Обсяг 1	1	2	3	4	5	5	6	7
Обсяг 2	2	3	3	4	5	6	6	7
Обсяг 3	2	3	0	5	6	7	8	9

За допомогою алгоритму визначення покриття безпеки семантичних БД були визначені рівні безпеки всіх триплетів метаданих. Отримані значення рівнів безпеки використовувалися для виконання контролю результатів при виконанні прямого запиту користувачів до семантичної БД.

На рис. 5.3 показані результати виконання прямого запиту до СБД користувачами, що мають різні рівні доступу $sl_1 = 2$.

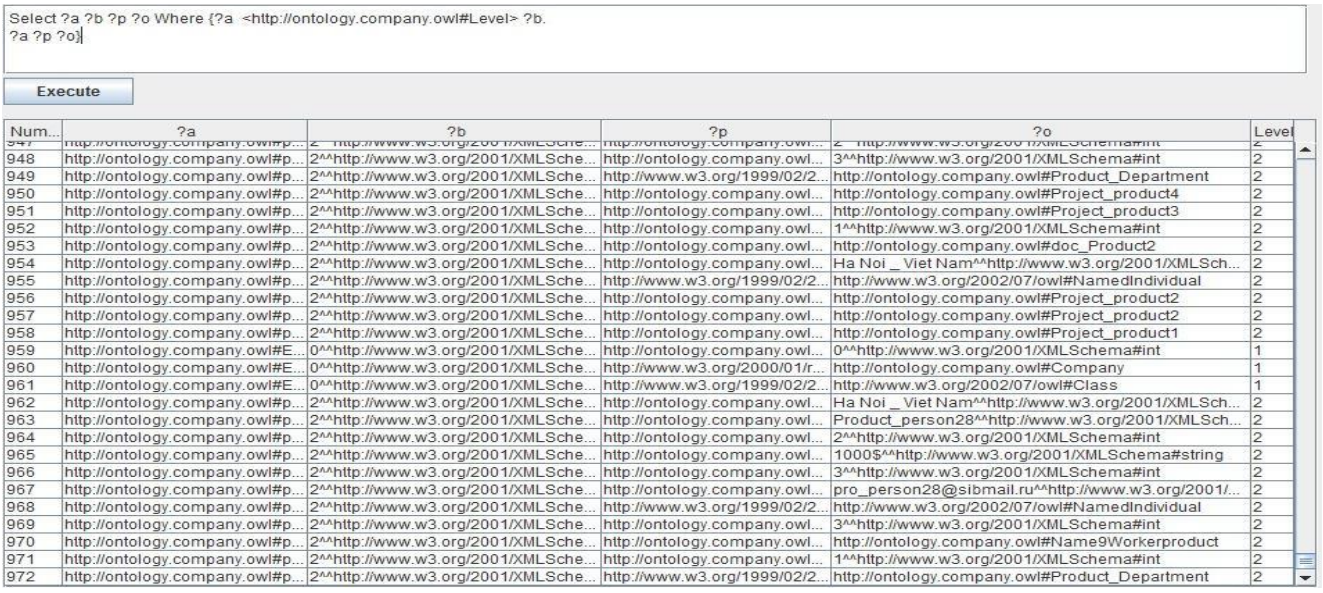


Рисунок 5.3 – Результат виконання прямого запиту користувачів, що мають рівень доступу $sl_1 = 2$

У результаті виконання запитів (див. рис. 5.3) показане, що користувачі, що мають рівень доступу $sl_1 = 2$, отримують тільки результати, що мають рівні безпеки не більш ніж 2.

Таким чином, можна зробити висновок про те, що алгоритми контролю виконання прямого запиту до семантичних БД гарантують, що користувачі отримують триплети відповідно до їхніх рівнів доступу.

У семантичних БД компанії «РІІТ» зберігається множина онтологій і метадані 16 профілів співробітників і 546 документів. Загальна кількість понять склало 162, кількість відносин – 137, кількість триплетів – 0.23 тисячи.

Основними онтологіями в семантичної БД компанії є онтологія користувачів, онтологія ресурсів, онтологія предметних областей. На рис. 5.4 показана частина онтології предметної області компанії.

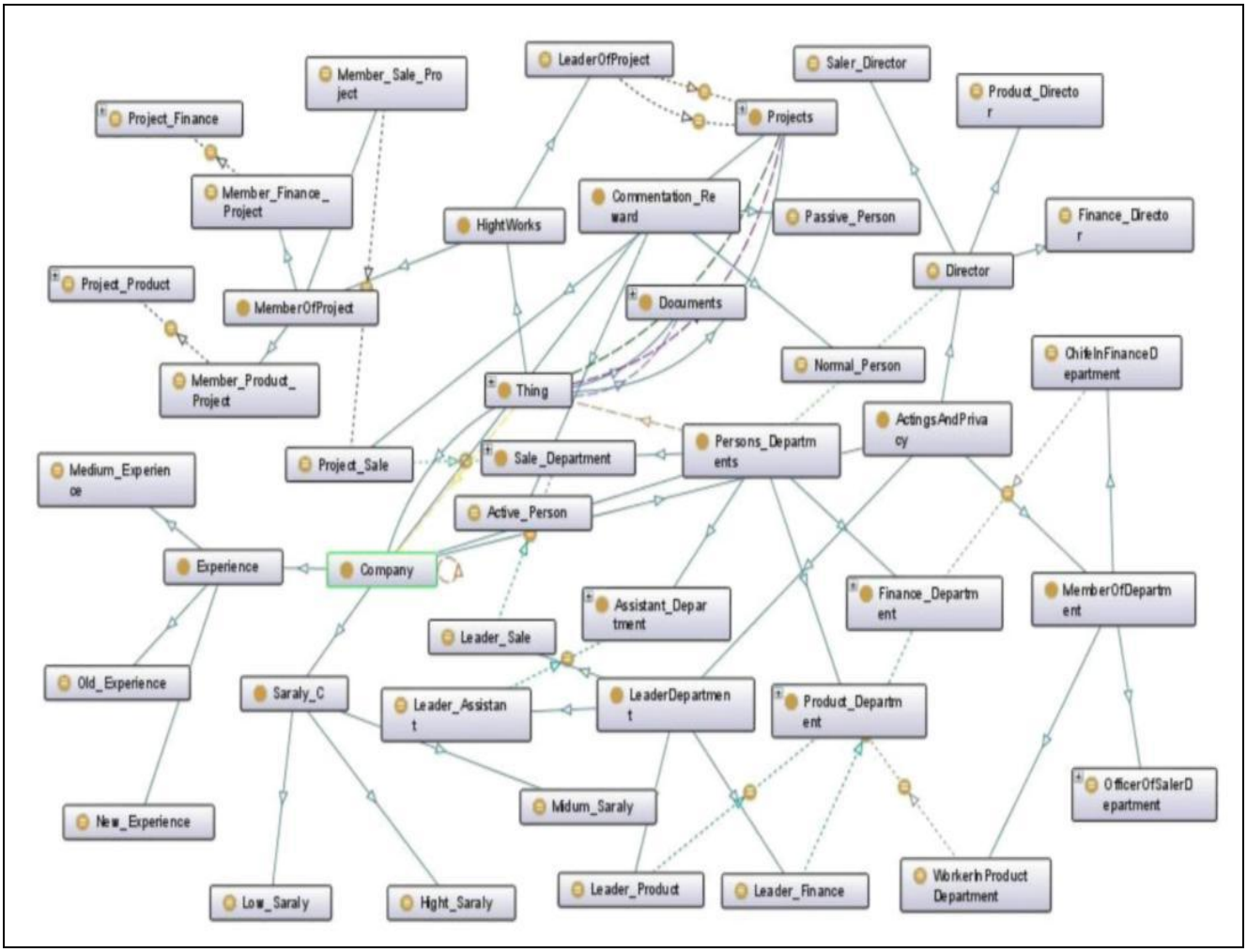


Рисунок 5.4 – Онтологія предметної області компанії РІІТ

При роботі з семантичної БД компанії розроблена підсистема дозволяє

контролювати доступ користувачів до даних і контролювати результати логічних висновків (див. рисунок 5.5).

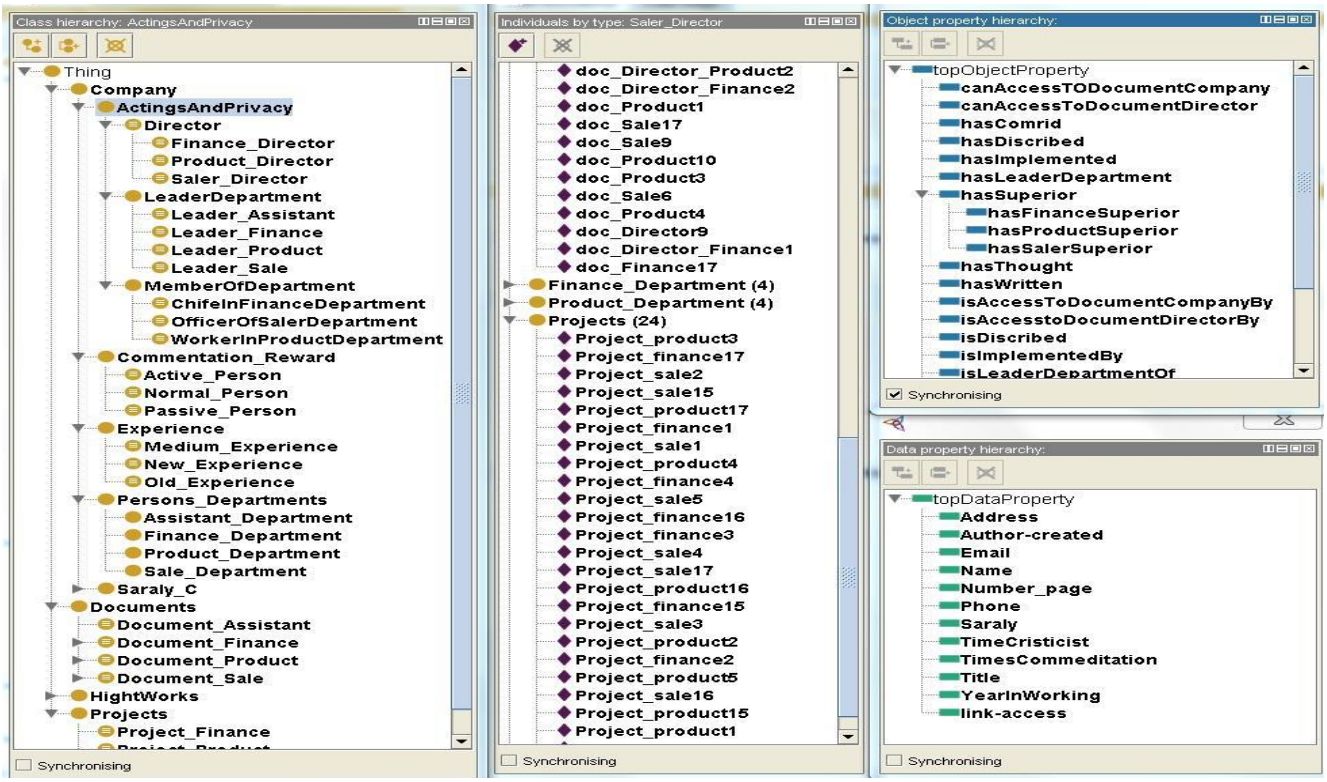


Рисунок 5.5 – Елементи онтології предметної області компанії PIIT

Вона дозволяє:

– задавати рівні безпеки елементів і триплетів в електронній бібліотеці;

– створювати і підтримувати профілі облікових записів користувачів;

– створювати логічні правила в підсистемі;

– контролювати виконання SPARQL-запитів з урахуванням прав доступу користувачів;

– контролювати доступ користувачів до елементам СБД і отриманої інформації для запобігання перевищення їх прав;

– контролювати результати, отримані при виконанні прямих і логічних SPARQL-запитів до семантичним БД;

– контролювати результати логічних висновків, що доступні користувачам.

Рівень абстрактних моделей об'єктів включає описи всіх об'єктів, процесів, провайдерів до різних джерел даних і сервісів системи за допомогою набору інтерфейсу, класів.

Рівень забезпечення безпеки даних дозволяє контролювати доступ користувачів до окремих елементів даних, результати логічних висновків.

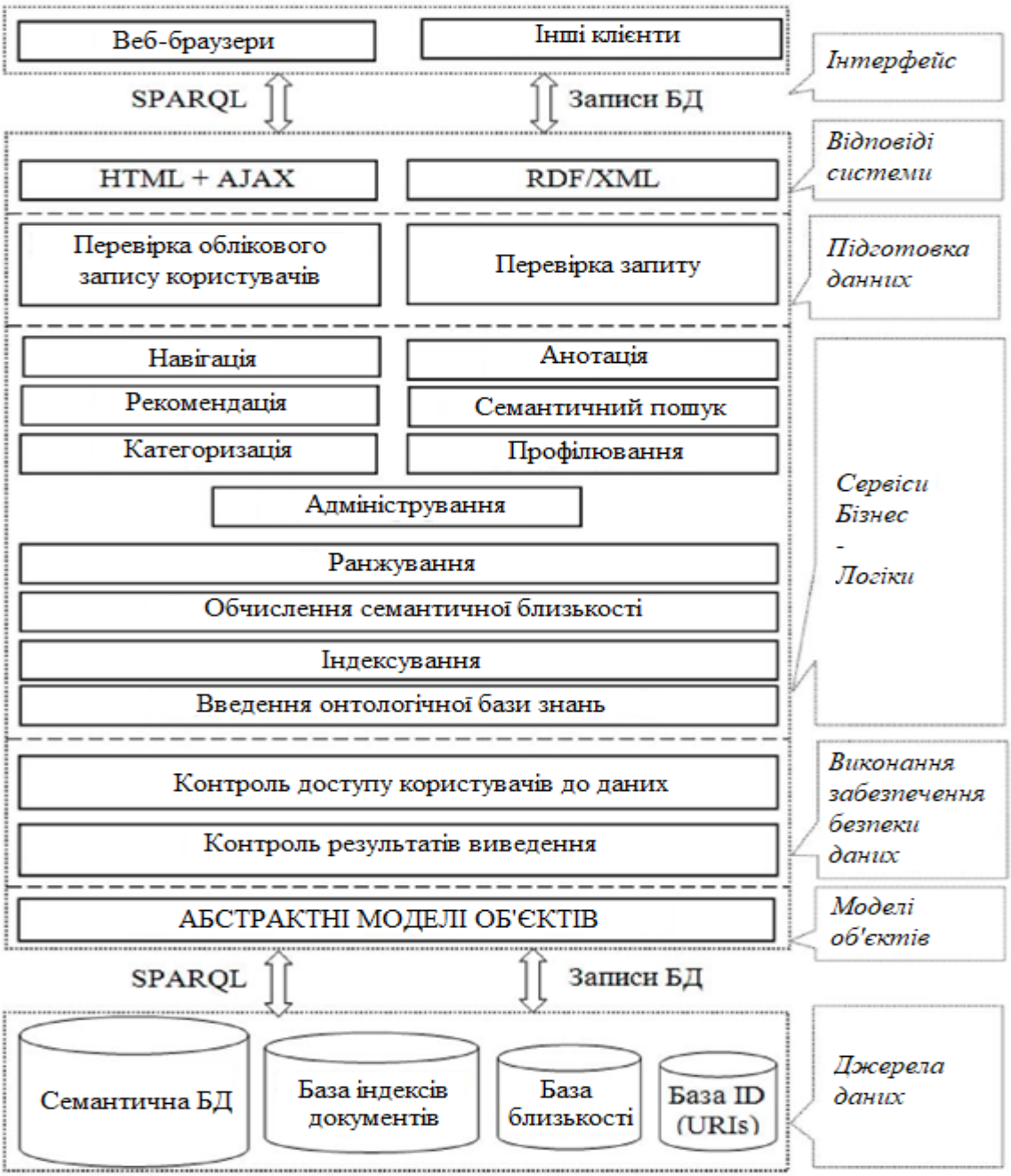


Рисунок 5.6 – Загальна архітектура семантичної електронної бібліотеки інформаційних ресурсів підприємства

Рівень джерел даних семантичної електронної бібліотеки включає семантичну БД і бази індексів документів, URI-ідентифікаторів об'єктів знань і попередні оцінки їх семантичної близькості. Як RDF-сховище обрана система Virtuoso Universal Server.

У семантичних БД компанії було завантажено набір онтологій, а також метадані 15 профілів співробітників і 850 документів. Загальна кількість понятій онтологій склало 169, кількість відносин – 217, кількість триплетів – 0.57 тисяч.

Основними онтологіями в семантичної БД компанії є онтологія користувачів, онтологія ресурсів, онтологія системи і онтологія предметної області.

На рис. 5.7 показана частина онтології предметної області.

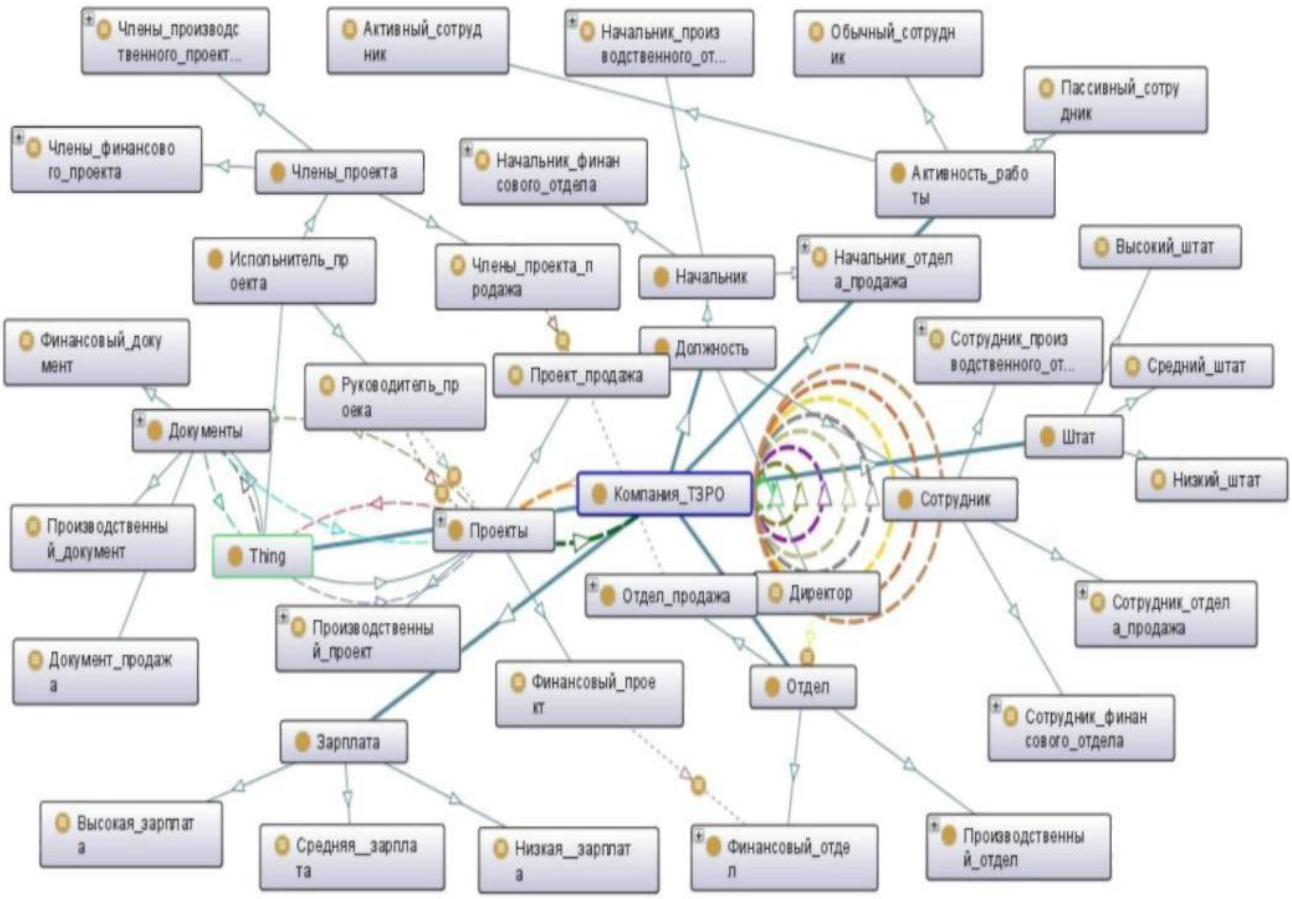


Рисунок 5.7 – Частина онтології предметної області компанії

На рівні забезпечення безпеки даних була реалізована інформаційна підсистема «Забезпечення безпеки семантичних баз даних ресурсів підприємства», в якій були використані методи і алгоритми, створені в даній роботі. Ця підсистема включає наступну функціональність:

– завдання рівнів доступу користувачів і рівні безпеки триплетів в БД;

– виконання узгодження рівнів безпеки всіх елементів онтологій, що зберігаються в БД;

– контроль доступу користувачів до елементам БД;

– виявлення порушень результатів логічних висновків;

– виконання SPARQL-запитів з урахуванням прав доступу користувачів;

– редагування (видалення, додавання та модифікування) триплетів семантичних даних в залежності від прав доступу користувачів;

Розроблена програма дозволяє контролювати результати логічних правил.

Розроблена загальна архітектура системи контролю доступу користувачів і контролю результатів логічних висновків в семантичних БД дозволяє забезпечити надійну безпеку роботи користувачів з семантичними БД.

Час виконання алгоритмів визначення рівнів безпеки триплетів семантичних БД істотно залежить від кількості понять онтологій, і незначно залежить від кількості триплетів.

Час виконання алгоритму визначення рівнів безпеки результатів логічних висновків також істотно залежить від кількості понять онтологій і незначно залежить від кількості триплетів в СБД.

Алгоритм виявлення порушень результатів логічних висновків і алгорітм контролю отриманих результатів при виконанні запитів дозволяють контролювати результати логічних висновків в семантичних БД.

ВИСНОВКИ

В рамках виконання кваліфікаційної роботи магістра сформовані теоретичні та практичні основи для вирішення завдання підтримки безпеки роботи з семантичними БД. До основних отриманих результатів належать такі:

Розроблено алгоритми узгодження рівнів безпеки елементів семантичних БД, що дозволяють визначити рівні безпеки класів, властивостей онтологій та індивідів метаданих. Розроблено алгоритм визначення покриття безпеки семантичних БД, що дозволяє визначити рівні безпеки всіх триплетів.

Розроблено алгоритм визначення покриття безпеки результатів логічних висновків, що дозволяє визначити рівні безпеки всіх результатів логічних висновків, отриманих шляхом використання логічних правил в семантичних БД.

Створено метод виявлення порушень результатів логічних висновків в семантичних БД, що виявляє всі можливі порушення безпеки результатов логічних висновків.

Розроблено алгоритм контролю отриманих результатів при виконанні запитів до семантичним БД, що гарантує отримання відповідей на запити користувачів відповідно до їх рівнями безпеки.

Запропоновано алгоритм забезпечення безпеки семантичних БД, що дозволяє з допустимими затримками забезпечити підтримку безпеки СБД при інтенсивному навантаженні.

На основі виконаних обчислювальних експериментів доведена ефективність розроблених алгоритмів, що використовуються для створення системи забезпечення безпеки семантичних БД.

Розроблено програмне забезпечення для контролю доступу користувачів до семантичним БД і результатів логічних висновків в семантичних БД длля ІТ-фірми «РІІТ», м. Харків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kienast R. Semantic Data Integration on Biomedical Data using Semantic Web Technology / R. Kienast, C. Baumgartner // Trends and Methodologies. – 2011. – P. 57–76.

2. Hendler A. J. Handbook of Semantic Web Technologies. – Springer, 2011. – 479p.

3. Semantic Web Technologies in Automotive Repair and Diagnostic // URL: http://www.w3.org/2001/ sw/sweo/public/UseCases/Renault/.

4. RIF basic logic dialect // URL:http://www.w3.org/TR/2013/REC-rif-bld-20130205/.

5. Gruber T. Collective Knowledge Systems: Where the Social Web meets the Semantic Web // Journal of Web Semantics. – 2018. – V. 6, № 1. – P. 4–13.

6. The Description Logic Handbook / F. Baader, D. Calvanese, D. Guinness, D. Nardi, P. Schneider. – Cambridge University Press, 2017. – 574 p.

7. OWL Web Ontology Language Semantics and Abstract Syntax // URL: http://www.w3.org/TR/2004/REC-owl-semantics-20040210/

8. Babenko A., Lempitsky V. Aggregating Deep Convolutional Features for Image Retrieval // Proceedings of the IEEE International Conference on Computer Vision. – 2018. P. 1269–1277.

9. Berclaz J., Fleuret F., Fua P. Robust people tracking with global trajectory Computer Vision and Pattern Recognition. – 2016. – P. 744–750.

10. Shubin, I., Snisar, S., Zhyrnov, V., Slavhorodskyi, V.// Practical Application of Formal Representation of Information for Intelligent Radar Systems 2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings, 2019, pp. 433-436, 8632103

11. Drayer B., Brox T. Object Detection, Tracking, and Motion Segmentation for Object-level Video Segmentation // arxiv.org. 2016. – URL: https://arxiv.org/abs/1608.03066.

12. Hinton G. A practical guide to training restricted Boltzmann machines // Momentum. – 2010. – № 9(1).

13. Kim C., Li F. Multiple Hypothesis Tracking Revisited // Proceedings of the IEEE International Conference on Computer Vision. – 2019.

14. Konev A., Chigorin A., Krivovyaz G., Velizhev A., Konushin A. Traffic signs recognition on images with training on synthetic data // Technical vision in computer systems. – 2019. P. 65-66.

15. Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Kar- pathy A., Khosla A., Bernstein M., Berg A.C., Fei-Fei L. Imagenet large scale visual recognition challenge // IJCV. – 2015

16. Ruta A. A New Approach for In-Vehicle Camera Traffic Sign Detection and Recognition // IAPR Conference on Machine vision Applications (MVA). – 2009. – P. 509-513.

17. Аведьян Є.Д., Галушкин А.І., Селиванов С.А. Порівняльний аналіз структур полносвязных і сверточных нейронних мереж і їх алгоритмів навчання // Інформатизація й зв'язок. – 2017. – № 1.

18. Антощук С.Г. Відстеження об'єктів інтересу при побудові автоматизированных систем відеоспостереження за людьми // Електротехнічні й комп'ютерні системи. – 2018. – №8(84). – С. 151–156.

19. Zhai M., Roshtkhari M., Mori G. Deep Learning of Appearance Models for Online Object Tracking // arxiv.org . 2019. – URL: https://arxiv.org/abs/1607.02568 .

20. Zhang K., Liu Q. Robust Visual Tracking via Convolutional Networks // arxiv.org . 2019. – URL: https://arxiv.org/abs/1501.04505 .

21. Zheng L., Bie Z. MARS: A Video Benchmark for Large-Scale Person Re-identification // Proc. European Conference on Computer Vision (ECCV). – 2016.

22. Xiang Y., Alahi A. Learning to Track: Online Multi-Object Tracking by Deci- sion Making // Proceedings of the IEEE International Conference on Computer Vision. – 2015.

23. Chetverikov G., Puzik O., Vechirska I. Multiple-valued structures of intellectual systems //Proceedings of the with Internations Computer Sciences and

Information Technologies (CSIT). 2016, 7589907. -pp. 204-207

24. Yang M., Wu Y. A Hybrid Data Association Framework for Robust Online Multi-Object Tracking // arxiv.org 2017. – URL: https://arxiv.org/abs/1703.10764 .

25. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition // Proceedings of the Neural Information Processing Sys- tems conference, NIPS. – 2015.

26. Szegedy C., Liu W., Jia Y. Going deeper with convolutions // CVPR. – 2015.

27. Talukder K.H., Harada K. Haar Wavelet Based Approach for Image Compres- sion and Quality Assessment of Compressed Image // IAENG International Journal of Applied Mathematics. – 2007. – 36(1).

28. Viola P., Jones M. Robust Real-Time Face Detection // International Journal of Computer Vision. – 2014. – V. 57. – №2. – P. 137–154.

29. A Neural Network for Machine Translation, at Production Scale. URL: https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

30. Добро пожаловать в блог DeepL! URL: https://www.deepl.com/blog/20180215.html

31. Shostak I., Matyushenko I., Romanenkov Yu., Danova M., Kuznetsova Yu. Computer Support for Decision-Making on Defining the Strategy of Green IT Development at the State Level. In book: Green-IT Engineering: Social, Business and Industrial Applications, Vol. 171. Berlin, Heidelberg: Springer International Publishing, 533–559 (2018), https://doi.org/10.1007/978-3-030-00253-4

32. Shostak I., Kapitan R,. Volobuyeva L., and Danova M., Ontological Approach to the Construction of Multi-Agent Systems for the Maintenance Supporting Processes of Production Equipment. In Proc. : IEEE International Scientific and Practical Conference «Problems of Infocommunications. Science and Technology» (PICS&T-2018). Ukraine, Kharkiv, October 9-12, 2018. P. 209 – 214

33. Кукіер, К. Big Data: A Revolution That Will Transform How We Live, Work, and Think/К. Кукіер, В. Штойнберг, 2018. – 236 с.