УДК 004.[832+75]

ТЕЛЕНИК С.Ф.,
АМОНС О.А.,
ЄФРЕМОВ К.В.,
ЖУК С.В.

СЕМАНТИЧНА ІНТЕГРАЦІЯ РІЗНОРІДНИХ ІНФОРМАЦІЙНИХ РЕСУРСІВ

В статье рассмотрена проблема интеграции информационных ресурсов Мировой системы данных. Проанализированы особенности информационного обмена данными центров Мировой системы данных в контексте поддержки междисциплинарных исследований. Выполнен аналитический обзор систем и технологий интеграции разнородных информационных ресурсов, разработана их систематизация по комплексу признаков. Предложен общий подход к решению проблемы интеграции на основе web-ориентированных технологий, способа организации взаимодействия источников даннях с посредником и моделей дескриптивной логики в терминах OWL, RDF и SPARQL. Рассмотрены модели, используемые для реализации подхода.

The article considers the problem of information resources integration of World Data System. There were analyzed the features of the information exchange performed by Data Centers of World Data System in the context of interdisciplinary research support. An analytical review of systems and technologies of heterogeneous information resources integration was carried out, their classification by the traits complex was developed. There was proposed a general approach to the solving of the integration problem on the basis of web-technologies, on the basis of the method of interaction organization between data sources with the mediator, and descriptive logic models in terms of OWL, RDF and SPARQL. The models used for the approach implementation have been considered.

Вступ

Існуючі центри даних Світової системи даних надають користувачам доступ до накопичених за великий період неоднорідних даних і різноманітні засоби та інструменти їх оброблення і аналізу [1]. Однак користувачі змушені переборювати певні труднощі при спробах отримати потрібні дані, якщо вони розподілені між декількома джерелами і потребують опрацювання із застосуванням декількох засобів та інструментів їх оброблення і аналізу. Ця комплексна проблема досить умовно поділяється на інтеграцію даних і застосувань, оскільки зазначені складові в реальних інформаційних системах тісно пов'язані. Авторами в праці [2] запропонований підхід до інтеграції застосувань. Однак його реалізація вимагає вирішення проблеми інтеграції даних у випадках, коли застосування мають опрацьовувати дані із різних і часто гетерогенних джерел. Одним із ефективних рішень цієї проблеми може стати створення семантизованого каталогу даних, який надає доступ до віддалених джерел даних за принципом «єдиного вікна». Для реалізації цієї концепції і створення інформаційних технологій реальної інтеграції різнорідних інформаційних ресурсів необхідно вирішити низку проблем, які й складають предмет виконаних у статті досліджень.

Сутність і проблематика інтеграції джерел даних

За більш ніж 50 років системою Світових центрів даних (СЦД, World Data Centers – WDC) і Федерацією астрономічних і геофізичних служб аналізу даних (ФАГС, Federation of Astronomical and Geophysical data analysis Services – FAGS) були накопичені величезні масиви даних і інформації, набутий великий досвід з організації міжнародного обміну даними, створені і впроваджені інформаційні технології (ІТ) довгострокового зберігання, аналізу та оброблення даних.

Однак стрімкий розвиток ІТ на переломі тисячоліть змінив традиційні уявлення про інформаційні процеси. Хмарні обчислення і широке впровадження сервісного підходу сприяли формуванню ІТ-середовища, якому притаманні консолідація програмно-апаратних засобів, конвергентність сервісів, віртуалізація ресурсів. Понад 50 функціонуючих на сьогодні СЦД не повною мірою інтегрувалися в нове середовище. Їх потужні системи накопичення і оброблення інформації виявилися не цілком адекватними потребам науки, недостатньо гнучкими для їх використання в міждисциплінарних дослідженнях.

Тому перед створеною у 2008 р. новою між-

дисциплінарною структурою – Світовою системою даних (ССД, World Data System - WDS), яка інтегрує осередки СЦД і ФАГС, поставлене завдання впровадити новий, координований глобальний підхід до організації збереження, оброблення і обміну науковими даними та інформацією, який гарантує універсальний рівноправний доступ до якісних даних та інформації для досліджень, освіти та прийняття рішень. Науковому комітету, який координує діяльність ССД необхідно вирішити низку проблем, пов'язаних з уніфікацією форматів і протоколів передачі даних, організацією контролю якості наукових даних та інформації та ін. [1].

По суті мова йде про більш масштабні проблеми інтеграції джерел даних, раціональної організації збереження, оброблення і поширення даних, формування ІТ-спільноти, здатної виконувати складні міждисциплінарні дослідження. Стисло розглянемо зазначені проблеми, перш ніж зосередитися на проблемі інтеграції джерел.

Світові центри даних, як правило, мають дисциплінарну спрямованість. Їх інформаційно-довідкова діяльність (а це один із основних напрямів діяльності СЦД), пов'язана з підтримкою архівів і баз даних, створенням та актуалізацією інвентаризаційних каталогів, формуванням метаданих, наданням необхідних консультацій кінцевим користувачам, можлива лише за наявності в СЦД фахівців у відповідних наукових галузях. Так, СЦД з геоінформатики та сталого розвитку (СЦД-Україна) в Україні покликаний забезпечити доступ українському науковому співтовариству до глобальних інформаційних ресурсів Міжнародної ради з науки в галузі наук про Землю, планетарної та космічної фізики та відповідних суміжних дисциплін, а також забезпечувати збір і зберігання національних наукових даних з зазначених дисциплін та їх репрезентацію світовому співтовариству. На СЦД-Україна також покладено збір, оброблення та аналіз світових даних, необхідних для досліджень у сфері сталого розвитку.

Впровадження сервісного підходу в такій великій розподіленій системі має особливості і становить складну проблем. Етапи еволюції програмних сервісів надання даних у СЦД [3]:

1. Дані надавалися користувачу просто як великі файли, причому усі завдання з пошуку та оброблення даних у ньому користувач повинен був виконувати самостійно.

2. Збережені СЦД дані було перенесено в бази даних, а користувачеві було надано засоби для введення запиту на них. Такий підхід вимагав від користувача знання мови запитів відповідної СУБД та інформації про схему БД, до яких необхідно виконати запит.

3. Клієнт-серверні технології локальних мереж обумовили в СЦД появу «тонких» та «товстих» застосувань-клієнтів, які надавали користувачам форми для вибору даних. Вони інкапсулювали в собі засоби генерування запитів до джерела даних, забезпечували користувачеві дружній інтерфейс, але розроблення таких форм часто вимагало істотних трудовитрат, і в більшості випадків – індивідуального підходу до кожного джерела (або файлу) даних.

4. Форми вибору даних еволюціонували у фасетні форми пошуку даних, у яких форми пошуку даних автоматично генеруються для кожного джерела даних і можна використовувати додаткові описи для збереження метаданих кожного джерела.

5. Створення системи сервісів надання даних за принципом «єдиного вікна» для доступу до даних. Така організація передбачає інтеграцію джерел даних в єдину систему з урахуванням їх семантики. Важливі на етапах 1 – 4 каталоги посилань до засобів надання даних та їх метаописи, необхідні для структурування доступних джерел даних, з яких розпочинав взаємодію із сервісом надання даних користувач, перетворюються у невід'ємну частину власне сервісу надання даних. Втілення цієї ідеї породжує наукову проблему інтеграції інформаційних ресурсів різнорідних джерел.

Розглянемо проблему інтеграції та каталогізації інформаційних ресурсів більш прискіпливо, щоб виділити її задачі і особливості, які необхідно врахувати при їх розв'язанні. По-перше, вимагають вирішення задачі інтеграції даних і застосувань, які вже давно стоять перед багатьма організаціями, проте до останнього часу технологічні можливості в цій сфері були досить обмеженими. По-друге, оброблення запитів до розподілених джерел даних породжує фундаментальні задачі побудови формалізмів для метаописання джерел і формування запитів, їх розбору (парсингу), формування і оптимізації плану виконання запитів [4]. Вони виникають у будь-яких системах зберігання даних – розподілених, централізованих чи паралельних – оскільки оброблювач запитів має отримати запит, зрозуміти значення його параметрів, сформувати план виконання запиту, оптимізувати і вико-

нати його, щоб отримати потрібні користувачам і застосуванням дані.

Формальні засоби метаописання джерел і формування запитів звичайно визначаються на основі логічних, алгебраїчних і графічних формалізмів. Метаопис має зберігати необхідну для формулювання, парсингу, перетворення і оптимізації запиту інформацію. Для цього він має містити схему джерела (джерел) даних (наприклад, визначення таблиць, уявлень, типи даних і т.д.), схеми поділу (наприклад, інформацію про те, що глобальні таблиці були розділені і як вони можуть бути відновлені), фізичну інформацію (розташування розділів таблиць, структуру індексів і статистику для оцінювання ефективності виконання плану).

Поданий у зручному для користувача вигляді запит має бути обробленим і переведеним у внутрішнє представлення, наприклад, граф запиту, зручне для парсингу.

Формування плану виконання запиту, насамперед, передбачає його еквівалентне перетворення, спрямоване на ліквідацію надлишкових предикатів, спрощення виразів і усунення вкладених запитів. На виході отримуємо план, який визначає послідовність виконання підзапитів до джерел даних та стратегію їх об’єднання. А для розподілених систем визначається, які частини запиту будуть виконуватися на яких джерелах даних.

Оптимізація плану обумовлена необхідністю врахувати при його виконанні фізичний стан системи – розмір таблиць, індекси, швидкість процесора тощо. Вона передбачає уточнення індексів і методів (наприклад, хешування і сортування), які будуть використовуватися для виконання запиту і його операцій (наприклад, об'єднання і групування), і в якому порядку вони будуть застосовані. Також уточнюються вузли, на яких виконуються операції, остаточно формується послідовність виконання підзапитів до джерел даних і стратегія їх об’єднання. Звичайно плани подаються у вигляді дерев, вершинами яких є оператори (з'єднання, групування, сортування та ін.), а ребра вказують послідовність виконання операторів.

Оптимізація запитів у гетерогенній системі має бути універсальною і базуватися на оцінках «вартості» виконання запитів до джерел даних. Застосовуються такі стратегії оптимізації [4]:

1. Визначається узагальнена модель оцінювання вартості виконання запиту для всіх компонентів і її коефіцієнти корегуються для кожного компоненту за результатами виконання набору тестових запитів.

2. Визначається своя модель оцінювання вартості для кожного компоненту.

3. Вартість плану виконання запитів визначається на основі опрацювання статистичних даних моніторингу виконання компонентами запитів.

Найвищу точність оцінок вартості виконання запитів забезпечує другий підхід, реалізація якого найвитратніша.

Після оптимізації виникає задача трансформації плану виконання запиту у виконуваний план, яка може включати генерацію асемблероподібного коду для ефективного виконання окремих частин плану, наприклад предикатів і виразів, на джерелі даних.

Задача власне виконання запиту полягає у використанні загальних реалізацій для кожного оператора. Організація роботи сучасних підсистем виконання запитів базується на моделі ітератора, яка забезпечує реалізацію операторів і стандартний інтерфейс.

Загалом, виходячи із необхідності збереження дисциплінарності СЦД, для інтеграції їх різнорідних джерел даних найперспективнішим видається підхід, реалізація якого дозволить звертатися до даних, трансформуючи їх у сумісні формати у реальному часі і не зберігаючи в проміжних сховищах. Це дозволить доповнити діяльність СЦД широкими можливостями обміну даними і доступу кінцевих користувачів до ресурсів усіх СЦД, незалежно від їх розташування, форматів даних і програмних засобів аналізу і оброблення даних, що складе надійну основу для розвитку міждисциплінарних досліджень.

Але реалізація такого підходу вимагає розв’язання наведених вище задач.

Огляд існуючих рішень

Спроби вирішення проблеми інтеграції різнорідних джерел наукових даних у самому ж науковому середовищі робилися вже досить давно. Фактично ця проблема ініціалізована появою і широким впровадженням перших промислових СУБД. В реальних умовах формування розподілених систем оброблення інформації, важливу складову яких становили розподілені бази даних, були сформульовані ідеї багаторівневих архітектур, сформувалися уявлення про моделі даних і роль концептуального моделювання, проектування систем баз даних

як реалізацію міжрівневих відображень. По суті термінологічну основу інтеграції закладено у відомому всім представникам старшого покоління IT-фахівців звіту ANSI/X3/SPARC.

Щоб розібратися у величезному різноманітті існуючих систем, підходів, технологій і засобів інтеграції, визначимося із ознаками їх класифікації і критеріями оцінювання. Систематизуватимемо існуючі рішення за такими ознаками:

а) вид інтеграції джерел даних: EII (Enterprise Information Integration); ETL (Extract, Transform, Load); EAI (Enterprise Application Integration);

б) спосіб організації взаємодії джерел даних: з посередником; peer-to-peer (P2P);

в) місце розташування описів джерел даних (каталогу): на центральному вузлі; повної копії загального каталогу на кожному вузлі; зберігання на кожному вузлі каталогу тільки тих об'єктів, які на ньому зберігаються; гібридний варіант;

г) архітектура програмної реалізації інтегрованої інформаційної системи: традиційна; клієнт-серверна; web-орієнтована;

д) рівень описання і маніпулювання даними: синтаксичний; семантичний;

е) метамодель описання інтегрованих даних: базова (реляційна, об’єктно-орієнтована, об’єктна); моделі класу NoSQL; простір даних; моделі дескриптивної логіки (в термінах OWL, RDF); структуровані моделі на основі XML; гібридні (SQL/XML) та ін.

За критерії оцінювання систем інтеграції інформаційних ресурсів виберемо [5]: гетерогенність; розподіленість; автономність. За додаткові критерії їх оцінювання приймемо доступність, цілісність і ефективність.

В умовному координатному просторі зазначених критеріїв досить рел’єфно проявляється місце кожного з підходів до інтеграції, які, у свою чергу, характеризуються набором ознак. Стисло розглянемо особливості систем інтеграції, які закріплюємо за наведеними ознаками.

Розпочнемо з *виду* інтеграції, який визначається на основі праць [6, 7]. Реалізація систем EII, які використовуються для інтеграції даних в режимі реального часу, здійснюється на основі загального шлюзу (gateway) з єдиними мовою і точкою доступу до неузгоджених джерел даних. За рахунок віртуалізації вони надають застосуванням і користувачам гнучкий, незапланований доступ до даних з будь-якою періодичністю і обсягами.

Системи ETL, які використовуються для пакетної інтеграції, передбачають вилучення даних із гетерогенних джерел, їх трансформацію і завантаження в єдине сховище даних. Вони звичайно застосовуються як сховища добре документованих надійних даних для багатовимірного аналізу, наприклад, часових рядів. Системи цього класу забезпечують видалення дублювання, перевірку якості даних та інші процеси обслуговування співробітників організацій і підтримки їх комплексів бізнес-процесів.

Системи EAI використовуються для інтеграції програм. Перетворення даних виконуються застосуваннями, за рахунок координованої взаємодії яких і забезпечується інтеграція. Традиційно системи цього класу підтримують взаємодію застосувань в реальному часі для автоматизації бізнес-процесів. Вони забезпечують узгодженість при внесенні змін у застосування.

Важливим для розуміння сутності інтеграції джерел даних є їх поділ за *способом організації взаємодії джерел даних*. Для систем інтеграції з посередником, які базуються на концепції медіатора [4], клієнти підключаються до mediator-компонентів (посередників) як показано на рис. 1. Медіатор здійснює розбір, переписування та оптимізацію запиту, виконує деякі з його операторів. Для цього використовується каталог, у якому зберігається глобальна схема гетерогенної системи баз даних, встановлюється відповідність її частин кожному компоненту бази даних. На основі каталогу формулюються запити прикладних програм і користувачів, зовнішніх компонентів, оптимізуються запити.

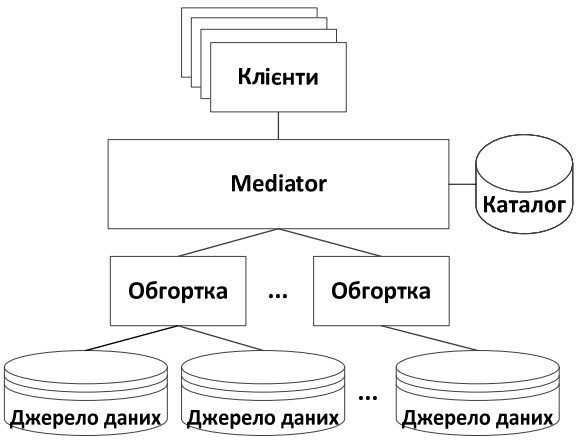

Рис. 1. Загальна схема системи з посередником

Щоб об’єднати деталі компонентів бази даних, створюються обгортки («wrapper», «адаптер»), пов'язані з кожним компонентом бази даних. Обгортки транслюють кожен запит ме-

діатора, щоб він був зрозумілий цільовій базі даних, і одержувані від бази даних результати, щоб вони були зрозумілі медіатору і сумісні з зовнішньою схемою бази даних і глобальною схемою гетерогенних баз даних. Створення обгорток не становить складну проблему.

Розширюваність забезпечується тим, що обгортки та компоненти бази даних можуть бути оновлені або нові компоненти баз даних можуть бути інтегровані без зміни посередника та існуючих обгорток.

Реалізації на основі медіаторів, насамперед, DISCO, Garlic, IBM Data Joiner, інструментів для побудови розподілених систем баз даних продемонстрували їх високу ефективність, особливо для інтеграції реляційних і об'єктно-орієнтованих баз даних. Ці інструменти вдало використовують переваги трирівневих архітектур програмних систем і видаються дуже зручними для інтеграції розподілених, насамперед, семантизованих джерел даних.

Структуру систем інтеграції даних на основі взаємодії P2P [8] складають автономні вузли (джерела даних), пов'язані між собою за рахунок відображень. Кожний з них надає свою частину інформації, доступної з розподіленого середовища. Цим децентралізованим системам притаманні гнучкість, здатність обробляти динамічні зміни в системі, оскільки вузли можуть підключатися або відключатися, не перериваючи роботу всієї системи.

Важливо розуміти, що в цих системах кожний вузол є системою інтеграції даних з медіатором, управляючи локально пов'язаними джерелами даних (локальний мапінг) за рахунок трансформованих в глобальну вузлову схему локальних відображень (глобальний мапінг). Вузли запрошують інформацію у будь-якого вузла (від зовнішніх користувачів або інших вузлів) на основі відображень, які зберігаються в їх специфікаціях. Опитуваний вузол завдяки мапінгу для формування відповіді може використовувати дані інших вузлів. Покладаючись тільки на доступні у вузлах сервіси, вузли системи ефективно надають відповіді на запити в термінах їх схем даних. Конвертація усіх відповідей в єдиний формат з використанням відображення між вузлами може виявитися складною, як і обмін даними і відображеннями даних між вузлами.

Щодо переваг і недоліків систем інтеграції, виділених за ознакою *місця розташування описів джерел даних*. Загалом описи джерел даних (метаописи, каталог для збереження метаданих) містять інформацію, яка дозволяє одним вузлам звертатися до об'єктів баз даних, що містяться на інших вузлах. Перші два варіанти розташування незадовільні за критерієм автономності, оскільки порушують принцип незалежності вузлів від центрального вузла або частково усіх вузлів, адже будь-яке оновлення каталогу повинно виконатися на всіх вузлах системи. Третій варіант зберігання незадовільний за критерієм ефективності, оскільки пошук віддаленого об'єкта вимагатиме звернення в середньому до половини вузлів.

Існують ефективні реалізації поєднання цих підходів. Так, на кожному вузлі однієї з перших промислових федеративних СУБД IBM System R, зберігалася інформація про локальні об'єкти даного вузла і частина інформації загального каталогу – інформація про створені на цьому вузлі об'єкти. Складові системного імені кожного об'єкта однозначно ідентифікували його в федеративній базі даних [9].

Архітектура програмної реалізації інтегрованої інформаційної системи є визначальною для досягнення критеріїв інтеграції. Різноманітні успадковані традиційні архітектури – від файлових систем до пакетів програм – загалом мають незначні інтеграційні можливості і їх включення в множину джерел вимагає багатьох зусиль. Клієнт-серверні трирівневі архітектури з виділеними серверною і клієнтською частинами і сервером баз даних дуже зручні для інтеграції на синтаксичному рівні баз даних, особливо реляційних і об’єктно-реляційних і меншою мірою об’єктних. Web-орієнтовані архітектури зручні саме для інтеграції на семантичному рівні, оскільки мають стандартизовані засоби обміну даними і взаємодії в термінах онтологій.

Системи інтеграції з *синтаксичним і семантичним рівнями* описання і маніпулювати даними відрізняє низка особливостей, які обумовлюють їх переваги і недоліки [10]. Синтаксичні структура даних орієнтована на однорідні набори даних, тоді як семантичні – на зв’язки та відношення між одиницями даних без залежності від їх подібності. Якщо у перших на запит одних джерел вибираються дані з інших джерел і перетворюються у відповідності із заданими вимогами, то у других встановлюються зв’язки між одиницями даних у відповідності до визначень у їх онтологічних описах. До того ж, на відміну від синтаксичної інтеграції, зникає потреба у завантаженні даних у проміжні сховища.

При синтаксичній інтеграції вартість з кожним новим джерелом даних зростає експоненційно, тоді як при семантичній практично не залежить від кількості джерел, оскільки у цьому випадку потрібно лише підготувати описи нових джерел. Традиційні для синтаксичної інтеграції жорстка стандартизація та форматування, порушення яких призводить до втрати контексту, поступаються при семантичній інтеграції гнучкому дотриманню стандартів, можливості введення унікальних типів у межах кожного опису джерела даних.

Насамкінець розглянемо особливості систем інтеграції, виділених за ознакою *метамоделі описання інтегрованих даних*. Оскільки більшість з них широко відомі, стисло охарактеризуємо концепції семантичної інтеграції даних, побудовані на методах визначення семантики предметної області. Для інтеграції гетерогенних інформаційних ресурсів необхідно забезпечити інтероперабельність на семантичному рівні, що породжує задачі порівняння вмісту ресурсів, відшукання відповідностей і розв'язування конфліктів між ними, а також проблему сполучення неоднорідних ресурсів [11].

Повна специфікація ресурсу повинна охоплювати специфікації: власне джерела даних; обмежень цілісності; контексту (області, у якій передбачається використання ресурсу).

Структура і поведінка інформаційного ресурсу визначена семантикою його предметної області. Умовою ефективного використання семантики даних є подання їх специфікації деякою формальною мовою. Теоретичну основу такої мови може скласти мова числення предикатів першого порядку. Розвиток web-технологій сприяв появі мов представлення знань в недвозначному, формалізованому вигляді на основі дескриптивної логіки [12].

Як інструмент створення таких узагальнених специфікацій можна використати онтологічні специфікації, розуміючи під онтологією формалізований опис загальноприйнятого розуміння деякої предметної області, за допомогою якого можуть спілкуватися люди, комп'ютерні системи [13]. Для інформаційного компонента така специфікація становить набір визначень і понять, а також правил (аксіом), пов'язаних з визначеннями і поняттями предметної області.

Онтологія визначає загальний словник для спілкування з використанням інформації предметної області На відміну від метаданих, таких як тип, розмір атрибута, онтології повинні мати виразніші засоби визначення семантики даних, насамперед традиційну ієрархію понять і типів об'єктів, разом з точним описом кожного типу, аксіоми, які задають обмеження на інтерпретації цих понять.

Розроблення сучасних програмних систем, а надто їх інтеграція, має базуватися на стандартних онтологіях, які дозволяють спільно використовувати інформацію відповідних предметних областей людьми і комп'ютерами. Уже розроблено багато онтологій, що уможливлює повторне використання, аналіз і класифікацію знань, явне формулювання припущень в предметній області.

Оскільки розроблення онтологій полягає у визначенні набору даних та їх структури для використання іншими програмами, для кодування онтологій необхідна формальна мова — мова опису онтологій. Обмін програм та програмних агентів даними, описаними в термінах онтологій, дозволить забезпечити їх реальну взаємодію, незалежність від форматів представлення даних. Потреба в онтологічних моделях на основі логік стимулюється реалізацією Semantic Web — нової концепції розвитку Інтернет, підтримуваної World Wide Web Consortium (W3C). Світовою IT-спільнотою розроблено багато мов описання онтологій, наприклад Knowledge Interchange Format, Common Logic, Cycl, Simple HTML Ontology Extension (SHOE), Ontology Markup Language (OML). Вони забезпечують подання зрозумілого для комп'ютерів семантичного знання в документах Web, дозволяючи програмам і програмним агентам збирати значиму інформацію на Web-сторінках і документах, поліпшуючи механізми пошуку і опрацювання знань. Ядерна частина цих мов пов'язана з логічними аспектами, а онтологія визначається як структура, компоненти якої охоплюють декілька рівнів представлень, наприклад концептів, об'єктів, відношень, функцій, аксіоми і онтологій, використовуючи концептуальні графічні можливості. Найбільш поширені такі основні мови:

1. XML (eXtensible Markup Language) — метамова, розроблена W3C для реалізації web-орієнтованих програмних систем і інтероперабельності з Standard General Markup Language (SGML) та HTML. Вона має зручний для аналізу програмами і людського сприйняття синтаксис, який дозволяє визначати складні структури даних, опрацьовувати документи. Це перетворює її у простий і потужний спосіб визначення

мов специфікацій онтологій і засіб обміну онтологіями [14];

2. DARPA Agent Markup Language + Ontology Interchange Language (DAML+OIL) — сформований на стандартах W3C результат злиття мов DAML і OIL, розроблений для надання семантичної інтероперабельності в XML. Вона має розширений набір потужних примітивів моделювання і використовується для формування онтологій з формальною семантикою і сервісами логічного висновку на основі дескриптивної логіки, обміну онтологіями.

3. Ontology Web Language (OWL) [15] — мова описання Web-онтологій, розроблена W3C як розширення Resource Description Framework (RDF) і RDF Schema (RDFS). OWL забезпечує поряд з формальною семантикою додатковий словник, що закладено в основу DAML + OIL, а також вбудовану підтримку відображення онтології. Вона має три діалекти Lite, DL і Full, які відрізняються формальною складністю і відповідно виражальними можливостями: від класифікаційної ієрархії і простих обмежень до максимальної виразності і застосування аксіом і правил для логічного висновку і складними обмеженнями та синтаксичною свободою RDF без гарантій обчислення. На сьогодні вони забезпечують потреби створення програмних систем інтеграції і підтримку взаємодії застосувань і через них бізнес-процесів у семантичних термінах [16].

Результати огляду мов описання онтологій для Semantic Web дозволяють зробити висновок, що вони відповідають потребам створення програмних систем інтеграції. Залишається розглянути засоби семантичного описання інформаційних ресурсів та виконання запитів до них.

Для описання інформаційних ресурсів використовується рекомендована W3C мова RDF [17]. Ця мова є абстрактною моделлю і описує ресурси у вигляді орієнтованого позначеного графа – кожний ресурс може мати властивості, які в свою чергу також можуть бути ресурсами або їх колекціями [18]. RDF, забезпечуючи універсальний, гнучкий метод декомпозиції знань на маленькі частини – триплети і враховуючи їх семантику, дозволяє задати структуру опису джерела (бази даних, XML-документу, Web-сайту тощо). При цьому можна використовувати і розширювати вбудовані поняття RDF-схем, такі як класи, властивості, типи, колекції, успадкування класів і властивостей. Крім опису структури, RDF дозволяє визначати і оперувати твердженнями на основі предикатів, що дозволяє подавати концептуальну інформацію.

Найбільш популярною серед мов запитів до RDF-сховищ на сьогоднішній день є мова SPARQL Protocol and RDF Query Language [19]. Вона визначає тільки запити до інформації, що міститься в моделях, а реалізацію інтерфейсу для мови запитів забезпечує бібліотека для розроблення семантичних веб-застосувань Jena [20]. SPARQL отримує вимоги застосування у формі запиту і повертає цю інформацію у вигляді набору зв'язків або графа RDF.

Тепер розглянемо існуючі рішення для інтеграції інформаційних ресурсів. Розпочнемо із засобів генерації і виконання семантичних запитів до розподілених джерел даних, які можуть надавати дані у форматах абстрактної моделі RDF.

Оброблювач ARQ запитів SPARQL для Jena надає розширення мови запитів для виконання запитів до розподілених джерел, розширюючи SPARQL для передавання шаблону триплету до віддаленої кінцевої точки доступу [20]. Це не є всеосяжним рішенням розподіленої обробки запитів, оскільки клієнт повинен знати джерело, до якого необхідно направляти запит, що неможливо при отриманні єдиного вхідного запиту, згенерованого автономним сервісом.

Проект DARQ пропонує єдиний інтерфейс для виконання запитів до довільної кількості розподілених SPARQL кінцевих точок і здійснює прозорий для клієнта поділ запиту на підзапити та їх виконання (федерування) [21]. Повністю побудований на SPARQL-стандарті, DARQ використовує сервіс-дескриптори для динамічного додавання і вилучення кінцевих точок, а також оптимізує запити на основі статистичної інформації.

Розвиток закладеного в DARQ підходу збагатив його новими ідеями. Так, в проекті SemWIQ [22] підтримуються запити до розподілених джерел даних у розширеному SPARQL-синтаксисі, підтримується робота з багатьма кінцевими точками з використанням конструкції COUNT. Оптимізація здійснюється на основі статистики про наповненість джерел даних, яка отримується моніторинговим компонентом в умовах наявності обмежень автономності, приватності та ін.

У праці [23] розроблено адаптивний алгоритм виконання запитів на SPARQL-кінцевих точках, який на відміну від статичних технік оптимізації застосовує динамічне об'єднання результатів виконання підзапитів лише за наявності зв'язуючих значень.

У проекті FedX [24] запропоновано ряд нових стратегій оброблення SPARQL-запитів до розподілених джерел даних на основі побудови під час виконання запиту єдиного об'єднаного RDF-графа. Якщо джерела даних надають свої SPARQL-кінцеві точки немає потреби у внесенні змін у їх структури даних.

Із реальних проектів варто згадати проект німецьких вчених PANGAEA під керівництвом Майкла Діпенброка [25], російський проект ЕСИМО (Єдина державна система інформації про обстановку в Світовому океані) [26] та ін. Так, перший із них присвячений створенню інформаційної системи PANGAEA за принципом бібліотеки відкритого доступу, призначеної для архівування, публікування і розповсюдження геоцентричних даних системних досліджень нашої планети. Система забезпечує довготерміновий доступ до контенту через інформаційні ресурси низки організацій. Кожний з наборів даних можна ідентифікувати, розділяти, публікувати і цитувати. Дані архівуються як додатки до публікацій або цитовані колекції даних. Цитування доступне через портал German National Library of Science and Technology.

Проект ЕСИМО присвячений створенню інформаційної системи для забезпечення морської діяльності Росії інформацією щодо ситуації у Світовому океані. Система автоматизує функції збору потрібних даних, їх збереженняі опрацювання з широким охопленням системних об’єктів, що перетворює її осередок міждисциплінарних досліджень.

Однак, жоден з цих програмних комплексів не може забезпечити повну інтеграцію різнорідних джерел даних. На жаль, часто недоступні вихідні коди розроблених рішень або в структурі їх даних не завжди може визначитися навіть досвідчений користувач. Крім того, ці проекти спрямовані на використання обмеженого набору моделей даних джерел. Для вирішення проблеми інтеграції потрібен більш гнучкий і універсальний підхід. Однак, справа ускладнюється тим, що існує декілька підходів-претендентів на провідне місце. Серед них досить перспективними видаються підхід, побудований на концепції просторів даних.

Ця концепція запропонована А.Гелеві [27], реалізована в ряді проектів, а в Україні розвивається Н.Б. Шаховською [28]. За умови створення конструктивного формалізму як основи визначення, пошуку і опрацювання даних проміжного шару можна сподіватися, що користувачі отримають ефективну і зручну технологію інтеграції. Але на сьогодні це лише загальна концепція, реалізація якої вимагає багато зусиль.

Отже, кожне з наведених рішень побудоване на цікавих ідеях, вирішує конкретні бізнесові або наукові проблеми. Вони оптимізовані під конкретну задачу, або навпаки, є настільки уніфікованими, що не можуть в повному обсязі бути використані для вирішення поставленої в цій роботі задачі. З іншого боку, всі оглянуті роботи по виконанню запитів до розподілених джерел даних не розраховані на взаємодію з не-RDF даними, не розглядають проблеми їх структурування та класифікації, що дуже важливо при роботі з інформаційними ресурсами Світової системи даних.

Постановка проблеми дослідження

В центрах Світової системи даних є багато формально не пов’язаних джерел даних, побудованих на базі різних інформаційних технологій – бази даних під управлінням СУБД, текстові файли, XML-документи, NoSQL-дані тощо. Необхідно, не вносячи зміни у структури джерел даних, що склалися історично, зробити їх доступними для користувачів за принципом «єдиного вікна». Одночасно необхідно забезпечити можливість виділення пов’язаних полів даних і виконання єдиного запиту до декількох джерел даних. Ще одним важливим аспектом проблеми є забезпечення можливості семантичного «розфарбування» існуючих даних, що зробить їх придатними для машинного оброблення із застосуванням технологій семантичного рівня і більш ефективного пошуку.

Загальний підхід до вирішення проблеми інтеграції джерел даних

Для описання специфікації семантики джерел даних обрані дескриптивні логіки. Як інструмент подання узагальнених специфікацій обрано онтологічні специфікації. Позначимо онтологічний клас і властивість в контексті семантики як C-клас і C-властивість відповідно.

Для реалізації системи інтеграції будемо застосовувати стек технологій Semantic Web – абстрактну модель даних у вигляді орієнтованих графів «суб'єкт-предикат-об'єкт» RDF, мову описання онтологій OWL і мову запитів до RDF-подібних даних SPARQL.

Для високорівневого описання доступних в системі предметних областей необхідно створити центральну онтологію-класифікатор, яка складається з ієрархії С-класів і не залежить від джерел даних. Центральна онтологія може бути розширена іншими онтологіями такої ж структури або парами онтологія-відображення. Кожна така пара забезпечує підключення для конкретного джерела даних. Для цього онтологія описує структуру і зв'язки його С-класів і С-властивостей, а відображення вказує співвідношення його С-класів і С-властивостей до реальної структури джерела даних, а також параметри підключення до нього. Таким чином, запит до джерела даних може формуватися в термінах С-класів та С-властивостей.

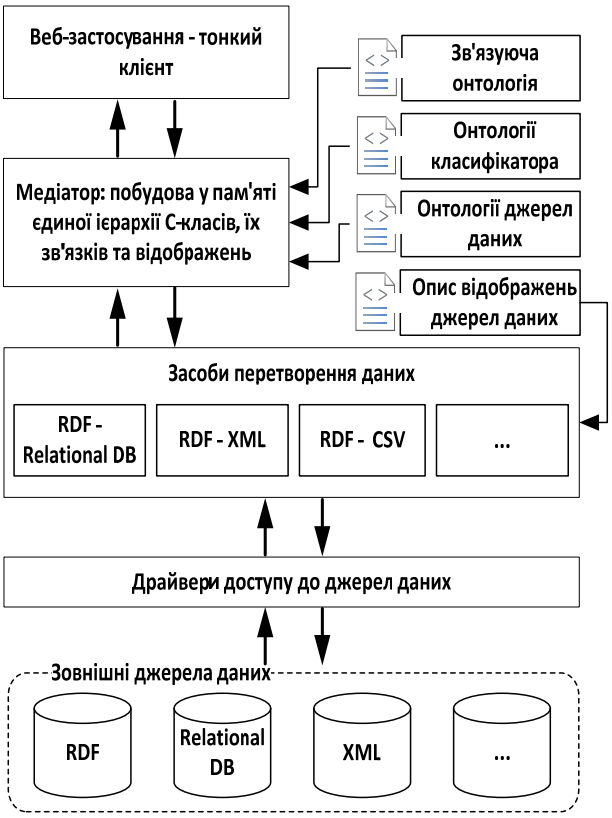

Рис. 2. Загальна схема рішення

Оскільки різні джерела даних можуть мати спільні С-властивості, створимо ще й зв'язуючу онтологію для встановлення зв'язку між такими С-властивостями різних джерел даних за допомогою стандартних предикатів мови OWL.

На рис. 2 наведено загальну схему запропонованого рішення інтеграції джерел даних класу EII з посередником і web-орієнтованою архітектурою. На боці клієнта вибір даних в каталозі, формування запиту до джерел даних, а також отримання відповіді на нього виконується за допомогою web-застосування. Медіатор є завантаженим в пам'ять сервера застосуванням і містить побудоване дерево С-класів з онтологічних специфікацій (саме їх користувач бачить як каталог), їх зв'язки із відображеннями на реальні джерела даних, а також, опціонально, отримані з онтологічних специфікацій та опитування джерел даних обмеження на допустимі діапазони значень даних.

Зв'язування онтологій на рівні застосування здійснюється за рахунок встановлення зв'язків їх С-класів з використанням унікального для кожного С-класу Uniform Resource Identifier (URI).

За прийнятими критеріями (гетерогенність, розподіленість і автономність) запропоноване рішення переважає рішення побудовані на множинних (multidatabase) і федеративних (federated database) базах даних, вже не кажучи про монолітні бази даних. Але за додатковими критеріями перевага не так відчутна.

Оброблення запитів до розподілених неоднорідних джерел даних. Концепція пропонованого рішення, схема оброблення запитів до розподілених джерел даних у якому наведена на рис. 3, позбавляє користувача від необхідності розуміння структур цих джерел, їх характеристик і особливостей мов запитів.

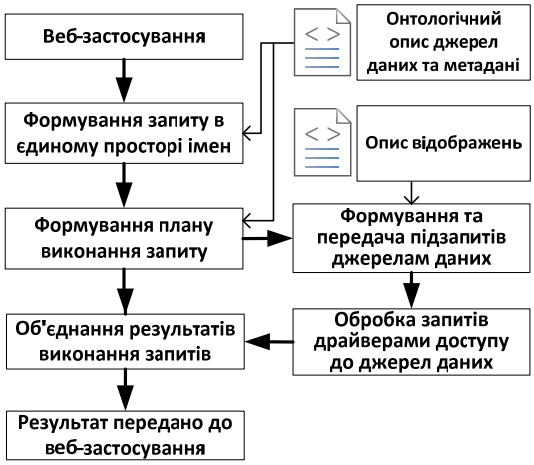

Рис. 3. Схема оброблення запитів

Точкою взаємодії з користувачем є web-застосування – тонкий клієнт. Він дозволяє користувачу формувати запит, оперуючи С-класами, С-властивостями, їх зв'язками та обмеженнями в глобальному просторі імен за допомогою меню-орієнтованого інтерфейсу. Інформація про структуру С-класів і С-властивостей, а також їх зв'язків завантажується з онтологічних описів джерел даних, а загальне дерево С-класів будується з онтології-класифікатора.

Далі застосування формує план виконання запиту, визначаючи, які джерела даних можуть

відповісти на запит або його частини, і чи можливо ці результати об'єднати. При цьому враховуються характеристики джерел даних, а також правила виведення, що визначають можливі операції над С-класами та їх С-властивостями, які отримані з онтологічних описів. Звідти ж береться інформація для формування умов-обмежень для джерел даних і запитів до них.

Після цього запит трансформується в єдиному просторі імен на основі описів відображень між джерелами даних. Він конвертується в запити до окремих джерел даних у відповідному синтаксисі, наприклад, у форматі SQL, і просторі імен.

Сформовані запити передаються у відповідні джерела даних за допомогою стандартних адаптерів, наприклад, JDBC. Обробка запитів адаптером доступу до джерела даних є стандартною практикою при роботі з СУБД. Джерела даних функціонують автономно. Після отримання запиту вони його виконують і повертають результат у вигляді набору записів.

Отримавши від джерел даних результати виконання запитів, застосування дотримуються раніше сформованого плану виконання запиту і виконують необхідні операції з ними – об'єднання, перетин, узагальнення, передача сервісам-обробникам і т.д.

Базова реалізація припускає, що всі джерела даних, застосовуючи стандартизовані адаптери, можуть обробляти SQL-або SPARQL-запити. Для взаємодії з джерелами даних, які надають XML-файли, або використовуються специфічні адаптери, або ці файли приводяться у формат онтологій з метаданими, описаними у XML-подібному синтаксисі. Перший підхід вимагає налаштувань і вартісних програмних продуктів, а другий – втручання у структури джерел даних на боці провайдерів. Наступні реалізації будуть охоплювати більш широке коло джерел.

Розглянемо модель для формування єдиного запиту до множини джерел даних користувачем у єдиному просторі імен. Користувач оперує ієрархією С-класів, зв'язками між ними (описані owl-предикатами SubclassOf і SameAs), а також С-властивостями, які можуть бути у кожного С-класу, і обмеженнями на ці С-властивості. На рівні логіки застосування відомо, які джерела даних пов'язані з якими С-класами і С-властивостями, але для користувача це не має ніякого значення.

Послідовність дій при формуванні запиту:

1. Вибір необхідного С-класу в дереві класифікатора.

2. Встановлення обмежень на значення його С-властивостей (за їх наявності) за допомогою констант або логічних виразів.

3. (Опціонально) встановлення зв'язків з С-властивостями інших С-класів з формуванням Join-запиту.

Таким чином, на виході може бути отримано один або більше одного С-класів, набір С-властивостей і обмежень на них, С-властивості, через які треба об'єднувати значення для отримання кінцевого результату. Виділимо чотири типи стандартних запитів до розподілених джерел даних, які можуть бути оброблені в межах цієї моделі:

1. *Запит до однієї сутності* – часто вживаний запит на отримання підмножини набору даних з одного джерела даних за деякою умовою. Класичним прикладом такого запиту є Select-подібний запит. У випадку запиту до даних з інтегрованою семантикою такий SPARQL-запит буде містити URI потрібного С-класу, за яким закріплене тільки одне джерело даних, і при цьому з інформації про діапазонах допустимих значень (при її наявності) випливає, що запит може бути виконаний на цьому джерелі.

2. *Запит, в якому сутність рознесена в декількох джерелах даних* (*Union*) – запит на отримання за деякою умовою даних результату об'єднання С-властивостей С-класів (які можна представити як таблиці) згідно їх специфікації з різних джерел даних. Тоді йдеться про вхідний запит з одним С-класом і умовою. При цьому у декількох джерелах даних відображено зазначений С-клас і виконується частково або повністю зазначена умова.

3. *Запит з об'єднанням сутностей* (*Join*) – запит на отримання перетину результатів кількох вибірок за С-властивостями різних С-класів, що може бути здійснено за наявності спільних С-властивостей у таких С-класів у зв'язуючій онтології. Зв'язки між зв'язуючими С-властивостями дозволять об'єднати запити для утворення в результаті об'єднаного за допомогою Inner Join набору значень.

4. *Запит на узагальнення* – запит на отримання з джерел даних наборів таблиць або описів С-класів, С-властивостей та їх зв'язків, які не підлягають об'єднанню, але, згідно опису в онтології-класифікаторі, вони є підкласами обраного користувачем С-класу. На практиці

отримання повних таблиць даних з усіх підкласів обраного С-класу може бути недоцільним і створити надмірне навантаження на потужності серверів-провайдерів, і це питання може потребувати окремого вивчення.

План виконання запиту та його оптимізація. Ефективність системи інтеграції даних визначається компонентом планування виконання запиту та його оптимізації. Оскільки джерела даних функціонують автономно, статистичні дані про їх наповнюваність отримати не завжди можливо. До того ж доцільність виконання надскладних запитів до підключених джерел даних, враховуючи їх специфіку у ССД і невелику кількість поєднуваних властивостей, є сумнівною.

Дієвим рішенням проблеми планування та оптимізації запитів є підхід на основі статистичної інформації, яку надають самі постачальники даних. Для неї виділено окреме поле метаданих в онтологічному описі джерела даних, яке заповнюється цілим числом, що вказує на орієнтовну кількість кортежів у джерелі даних в межах описуваної таблиці або описуваного перетину множини таблиць.

Таким чином, за необхідності побудови запиту двох і більше джерел даних, обробник генерує дерево-план виконання запиту, яке відпрацьовує запити до джерел з найменшою кількістю записів (рядків), передаючи в запиті до кожного наступного (пов'язаного з ним) джерела даних їх результат як параметр і одержуючи їх об'єднання на виході. За умови виконання запитів до більше, ніж одного джерела даних – вони виконуються паралельно. Тоді на кожному кроці відкидається певна кількість нерелевантних записів, як при SQL-запиті Inner Join, а на виході – отримуємо єдину таблицю із результатом виконання запиту.

Для побудови плану виконання запиту із врахуванням коефіцієнтів ємності джерел даних необхідно виконати такий алгоритм:

Крок 1. Отримання на вхід запиту у єдиному просторі імен, який визначає множину джерел даних і набір С-властивостей, за якими можуть бути об'єднані результати запитів до даних джерел даних, коефіцієнтів ємності кожного джерела даних;

Крок 2. Знаходження добутку коефіцієнтів ємності кожної пари джерел даних, підзапити до яких підлягають об'єднанню;

Крок 3. Виділення пари джерел даних, підзапити до яких підлягають об'єднанню;

Крок 4. Впорядкування цих пар у порядку зростання добутку коефіцієнтів ємності;

Крок 5. Виділення джерел даних, запити до яких можуть бути виконані паралельно (не потребують вхідних параметрів – результатів виконання запитів з інших джерел) та мають найменші коефіцієнти ємності;

Крок 6. Паралельне виконання запитів до виділених на кроці 5 джерел даних;

Крок 7. Виділення з отриманих на кроці 6 результатів даних, за якими виконується перетин із джерелом-парою та передавання як параметра в запиті до цього джерела-пари;

Крок 8. Представлення отриманого перетину результатів запитів до кожної пари джерел даних як віртуального джерела даних, коефіцієнт ємності якого дорівнює кількості кортежів отриманого результату, а запит вже було виконано;

Крок 9. Виконання кроків 3 – 8 доки не буде отримано єдиний кортеж, який і стане результатом виконання запиту користувача.

Формування запитів безпосередньо до джерел даних. Поля таблиць джерела даних приводяться у абстрактну модель даних у вигляді графу «суб'єкт-предикат-об'єкт» і навпаки за допомогою опису відображень (мапінг). Представимо опис відображення як

$$f: R \leftrightarrow S,$$

де R – вихідна структура джерела даних, S – абстрактне семантичне представлення у вигляді графу.

Для конвертації запитів, виділених до конкретних джерел даних, в SQL-формат з урахуванням властивостей і параметрів цих джерел згідно опису відображень використаємо алгоритм безпосередньої конвертації запитів, наведений у праці [29].

Побудова зв'язаної структури даних із онтологій та підключень до джерел даних. Важливою задачею є побудова у пам'яті застосування зв'язної ієрархії із онтології-класифікатора, розширюючих онтологій, онтологій джерел даних, онтологій, що зв'язуюсь С-властивості джерел, а також їх співставлення з описом відображення джерел даних. Для цього будемо використовувати такий алгоритм:

Крок 1. Завантаження онтології-класифікатора, побудова відповідного головного дерева С-класів.

Крок 2. Послідовне завантаження розширюючих онтологій з під'єднанням їх у пам'яті до головного дерева. Послідовність завантаження розширюючих онтологій може бути визначена

чергою (це необхідно, наприклад, для випадку, коли одна розширююча онтологія розширює іншу).

Крок 3. Послідовне завантаження онтологій-описів джерел даних. При цьому такі онтології можуть мати декілька рівнів підкласів, що робить їх порівнюваними з розширюючими онтологіями.

Крок 4. Приєднання завантажених онтологій-описів джерел даних до головного дерева, співставлення співпадаючих С-класів і їх С-властивостей та визначення їх (за наявності) як таких, що доступні у декількох джерелах даних.

Крок 5. Завантаження звя'зуючої онтології, проекція її на збудоване головне дерево С-класів і С-властивостей. Результатом є граф, який пов'язує деякі листки (С-властивості) збудованого головного дерева.

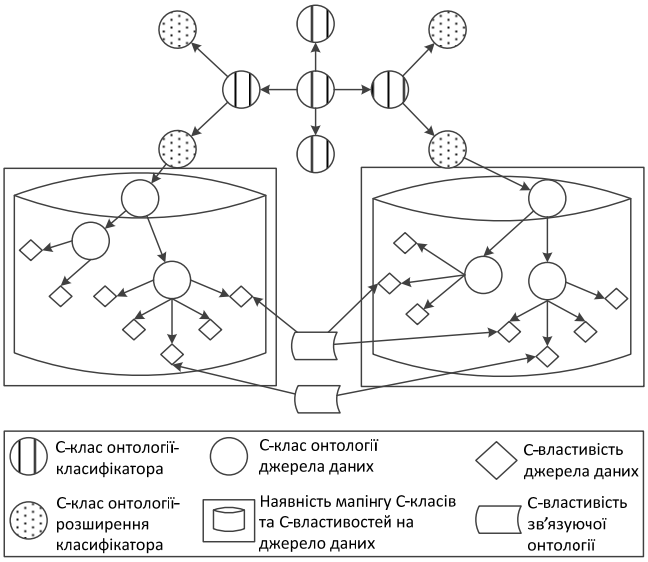

Рис. 4. Приклад об'єктної моделі об'єднаних онтологій та описів джерел даних

Крок 6. Завантаження описів відображень джерел даних і встановлення їх зв'язків з С-класами та С-властивостями.

Приклад побудованого в пам'яті застосування дерева С-класів, С-властивостей, їх зв'язків між собою та з джерелами даних наведений на рис. 4.

Структура і принципи функціонування системи. Структура системи визначається потребою в інтеграції багатьох джерел даних при розв'язанні міждисциплінарних наукових задач. Використання дескриптивних логік для описання джерел даних, модульність, готовність до приєднання ланцюжків сервісів для додаткового оброблення даних приводять до розподіленої системи з центральним вузлом, який працює на сервері застосувань як Java Enterprise компонент с налаштованою точкою доступу. Для приєднання кожного джерела даних добавляються OWL-файл з семантичною маскою метаданих і файл властвостей для безпосереднього підключення зовнішнього джерела з уточненням типу драйвера, IP-адреси тощо. Зрозумілою також стає роль користувацького інтерфейсу і системи конструювання запиту.

Компоненти системи функціонують на основі моделей, визначених нижче. Послідовність роботи компонентів:

1. Генерування SPARQL-запитів за допомогою конструктора або отримання готового запиту від користувача.
2. Парсинг, переписування і оптимізація запиту.
3. Разбиття запиту на підзапити.
4. Виконання підзапитів на вибраних джерелах даних.
5. Очікування отримання відповідей від усіх джерел даних.
6. Об'єднання отриманих результатів за допомогою Inner Join (опціонально).
7. Повернення користувачу єдиної таблиці або множини таблиць (якщо об'єднання неможливе або користувач вирішив його не виконувати) як результат виконання запиту.

Моделі і методи для реалізації окремих етапів

Моделі визначення семантики предметної області. Створення інформаційної системи для інтеграції неоднорідних інформаційних ресурсів, забезпечення їх семантичної інтероперабельності вимагає вирішення проблем порівняння вмісту цих ресурсів, встановлення відповідностей і запобігання конфліктів між ними, а також проблему сполучення різнорідних ресурсів. Як зазначалося вище, повна специфікація інформаційного ресурсу буде охоплювати специфікації власне джерела даних, обмежень цілісності і предметної області. Використання концепції медіатора вимагає описання структури і поведінка кожного інформаційного ресурсу. Вони визначаються семантикою відповідних предметних областей, специфікації якої описуватимемо на основі дескриптивної логіки. Її базові елементи:

- множина класів NC;
- множина індивідуумів NI;
- множина відношень NR.

Для описання інформаційних ресурсів будемо використовувати RDF і RDFS. Засновані на

ідеях представлення знань, таких як семантичні мережі, фрейми і логіка предикатів, прості в реалізації вони дозволяють визначати класи, ресурси і властивості, ієрархії й обмеження типів, концепти, відношення й екземпляри. Для визначення функцій і аксіом будемо використовувати інші мови.

Для описання онтологій будемо використовувати мову OWL, яка повністю відповідає вимогам до розроблюваних систем інтеграції інформаційних ресурсів.

Машина логічного висновку і перетворення даних у формат RDF. Розпочнемо з бібліотеки для організації мапінгу не RDF-даних на онтологію. Для інтеграції семантики предметної області з джерелом даних, необхідно реалізувати механізм мапінгу концептів онтології відповідним об'єктам джерела даних.

Таку задачу здатна здійснити бібліотека Java D2R, яка забезпечує доступ до даних через SPARQL, Jena API і Sesame API у вигляді RDF-графу. Бібліотека D2R забезпечує зв'язок з джерелом даних через JDBC драйвер, що дозволяє використовувати в якості джерела практично будь-який ресурс – від СУБД до плоских файлів і HTML-документів, а при необхідності реалізувати специфічний драйвер. Роль носія семантики предметної області відіграє OWL-онтологія. Файл мапінгу будується в вигляді RDF-документа в форматі N3, доступ до якого може бути реалізовано методами інструменту Jena, а отже – автоматизовано його створення.

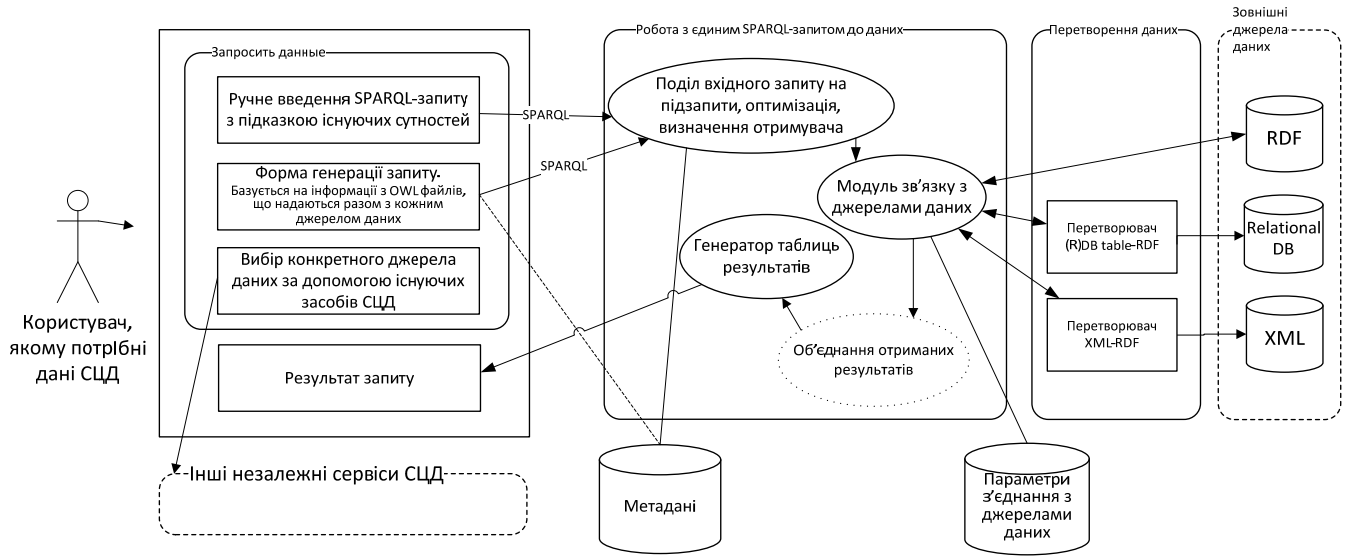

Рис. 5. Загальна схема

Іншим важливим аспектом створюваної системи є представлення користувачу єдиного доступу до метаданих, що включають семантику предметної області та інформацію з джерел даних. Зазначимо важливість простоти в роботі з клієнтською частиною системи. Зокрема, її реалізація в вигляді web-застосування виключить необхідність встановлення додаткового програмного забезпечення та його налаштування.

Наступним необхідним кроком є вибір засобу для створення і маніпулювання RDF-графами. Таким засобом вибрано Jena [30], що є Java API застосуванням. Jena має класи об'єктів для представлення графів, ресурсів, властивостей і літералів. Інтерфейси представлення ресурсів, властивості і літералів називаються відповідно Resource, Property та Literal. У Jena, граф називається моделлю і представлений інтерфейсом Model.

Таким чином, каркас Jena надає високорівневий доступ до графів RDF-документу і тим самим простий і гнучкий спосіб завантаження онтології, маніпулювання триплетами, що в ній містяться, об'єднання декількох графів, наприклад онтології та RDF-документу.

Для ефективного використання семантики предметної області для відповіді на запити користувача, необхідно реалізувати логічний висновок на базі побудованого графу.

Для розв'язання цієї задачі було вибрано машину логічного висновку Pellet [31], що є системою логічного висновку над RDF-даними з OWL DL (дескриптивна логіка). Ця система включає в себе оптимізацію для номіналів, відповіді на кон'юнктивні запити, а також послідовний висновок над новими графами.

Pellet підтримує висновок і надає функції перевірки цілісності, когерентності класу, класифікації для відповіді на запити, такі як отримання всіх або тільки прямих підкласів класу,

визначення прямого типу для індивідів. Важливо, що Pellet має API для спільної роботи з інструментом Jena.

Генерація єдиного запиту на семантизовані дані. Розглянемо генерацію первинного запиту на дані, сформульованого користувачем. Для генерування семантичних запитів використовуються такі підходи:

1. Користувач самостійно вводить запит на мові запитів SPARQL.

2. Запит конструюється за допомогою візуальних засобів, головною функцією яких є надання інтерфейсу для побудови семантичних графів-трійок «суб'єкт-предикат-об'єкт» і обмежень на них.

3. Користувач вводить запит природною мовою, а система надає засоби його розпізнавання і переведення в формат SPARQL.

4. Пошук за ключовим словом і його розвиток – інкрементарне конструювання запиту.

У запропонованій системі інтеграції використовується перший із цих підходів, але самостійне введення SPARQL-запиту до наявних джерел даних передбачає високий рівень компетентності користувача. Тому додатково розроблений наступний композитний алгоритм:

Крок 1. Користувач вибирає сутності і обмеження на них. Щоб уточнити цей крок, введемо необхідні позначення. Нехай $A = \{A_1,\ldots,A_m\}$ – множина доступних в системі сутностей в рамках глобального простору імен, $B = \{B_1,\ldots ,B_m\}$ – множина обмежень на діапазони даних в A_i, i=1,…,m, B_i, i=1,…,m, може бути порожнім, m — кількість доступних в системі сутностей в рамках глобального простору імен, $C \subset A$ — підмножина сутностей із множини A, яка визначається користувачем в процесі формування запиту і уточнює сутності, інформація про які має бути результатом пошуку.

Також можуть бути введені додаткові множини D, G, E (…) параметрів операторів опрацювання вибірки даних, наприклад LIMIT, ORDER BY, OFFSET і т.д.

Тоді запит на дані можна подати у вигляді кортежу:

<A[],B[],C[], *опціональні параметри*>

На практиці такий опис може бути реалізованим у вигляді веб-форми з вибором відповідних даних, а отримана із цієї форми інформація очевидним чином конвертується в SPARQL-запит, оскільки сутності кортежу є одночасно і сутностями цього запиту.

Крок 2. Оброблювач запитів отримує на вході запит, згенерований на кроці 1, і аналізує кількість джерел даних, в яких є інформація про потрібні сутності. Якщо таких джерел даних декілька, то користувачу пропонується уточнити запит, вибравши відповідне джерело даних. У цьому випадку користувач отримує стислу інформацію щодо кожного із можливих джерел даних для цього запиту. Ця інформація може витягуватися безпосередньо із джерела даних або із OWL-файлу зі стандартизованим форматом метаданих. Якщо користувач не може уточнити джерело даних, запит буде виконаний кожним можливим джерелом даних.

Моделі поділу запиту на розподілені дані на підзапити і його виконання

Саме ці моделі дозволяють побудувати ефективні алгоритми оброблення запитів до розподілених джерел даних, які, у свою чергу, визначають ефективність інтеграції джерел даних. За умови перетворення всіх імен даних у запиті у відповідності з глобальним простором імен для опрацювання єдиного запиту до розподілених джерел даних будемо використовувати описані в праці [4] рішення, які вже стали класичними.

Розроблені як частина проекту IBM Starburst вони закріплюють за оброблювачем, який отримав запит на вході три основних операції: аналіз і трансформацію запиту у план його виконання на основі наявної інформації щодо джерела (джерел) даних; багатокрокову оптимізацію запиту; виконання плану, щоб отримати результати запиту. Розглянемо моделі для кожної з наведених операцій.

Запропонований підхід до виконання запитів до розподілених джерел семантизованих даних узагальнює напрацьовані раніше рішення. Для виконання запитів до розподілених RDF- джерел даних, які отримують на вході SPARQL-запити, найбільш прийнятна система інтеграції D2RQ [29] і моделі, наведені у праці [32]. Система інтеграції D2RQ пропонує єдиний інтерфейс для опитування множини розподілених SPARQL-кінцевих точок і здійснює федерування запиту прозоро для клієнта. Виконувана D2RQ оптимізація SPARQL-запитів, побудована на статистичній інформації щодо кількості триплетів у джерелі даних, також досить ефективна. Але D2RQ погано документований і не здатний самостійно (без участі адміністратора) завантажувати інформацію щодо типів даних (дескрипторів) джерела, які мають специфічний формат.

Тому в запропонованій системі інтеграції впроваджені моделі федерування запитів до розподілених джерел даних, деякі алгоритми якої мають відмінності від розроблених для системи D2RQ:

1. Застосування як інформації щодо типів даних для кожного з підключених джерел даних файлів OWL і N3, які також застосовуються для уточнення семантики предметної області і додавання метаданих.

2. Забезпечення можливості виконання запитів не тільки на SPARQL-кінцевих точках, а й із застосуванням інших засобів комунікації, насамперед, обміну повідомленнями JMS або FIPA ACL.

3. Функціонування в умовах відсутності можливості оптимізувати запит, коли відповідні статистичні дані з віддалених джерел не доступні або змінилися.

4. Забезпечення можливості задавати цілий набір джерел даних, на які будуть направлятися згенеровані запити, що уможливлюється запропонованим алгоритмом генерації запиту, коли користувач уточнив доступні для виконання запиту джерела даних.

Федерування запиту на семантизовані дані. Розпочнемо з описання даних й адаптуємо класичну модель поділу, запропоновану в D2RQ. Ємністю джерела даних D є множина C_D кортежів $\boldsymbol{c} = (\boldsymbol{p}, \boldsymbol{r}) \in \boldsymbol{C_D}$, де p – наявний в D предикат, r – обмеження на суб'єкти і об'єкти. Ця зв'язка є регулярним filter-виразом мови SPARQL, яка робить вибір джерела даних більш точним, наприклад, можна уточнити, що джерело даних зберігає інформацію тільки про специфічні типи ресурсів. Позначимо обмеження r як функцію *r(subject, object)* з $\boldsymbol{r}: (\boldsymbol{RDF_T} \cup \boldsymbol{V}) \times (\boldsymbol{RDF_T} \cup \boldsymbol{V}) \to \{\boldsymbol{true}, \boldsymbol{false}\}$.

Застосовувати такі обмеження можна, наприклад, у тому випадку, коли в одному джерелі даних зберігаються рядкові дані, які починаються з букв від A до R, тоді як в іншому розташовані дані, які починаються з букв від Q до Z.

Обмеження на шаблони доступу. Деякі джерела даних мають обмеження на шаблони доступу. Наприклад, джерело даних може вимагати, щоб в запиті завжди були наведені назва міста, в якому виконується пошук, або його поштовий індекс.

Застосуємо рішення, запропоноване в [29]. Оскільки предикати повинні бути обмеженими, вони використані як основа для складання шаблону. Предикат не може бути змінною. Нехай L_D становить множину обмежень на шаблони доступу для джерела даних D і $(S, O) \in L_D$ буде одним шаблоном з S і O, які повинні мати зв'язані суб'єкти (S) або зв'язані об'єкти (O).

Джерело D може відповідати на запит з графовим шаблоном P, якщо він задовольняє щонайменше один із позначених шаблонів доступу для D. Нехай *bound(x)* буде функцією, яка повертає *false*, якщо x є змінною, і *true* в іншому випадку. Тоді шаблон доступу *(S, O)* задовольняється, якщо

$$\left(\forall p_s \in S \backslash O: \exists (s, p_s, o) \in P: bound(s)\right) \wedge \left(\forall p_o \in O \backslash \exists: (s, p_o, o) \in P: bound(o)\right) \wedge (\forall p_b \in S \cap O: \exists (s, p_b, o) \in P: bound(s) \wedge bound(o))$$

Статистична інформація. Наявність статистичної інформації щодо доступних даних допомагає оптимізатору запитів знайти план виконання запиту, ефективний за витратами. Але враховуючи незалежність і розподіленість джерел даних, збір статистичної інформації не завжди можливий.

Наповнення каталогу інформацією. На центральному вузлі системи (оброблювачі вхідних запитів) є каталог, який зберігає інформацію щодо всіх підключених сервісів.

Первинна інформація щодо даних у кожному джерелі вибирається із його опису – відповідних файлів N3 і OWL, які також використовуються для додавання семантичного забарвлення даним.

Планування виконання запиту. При формуванні запиту до розподілених джерел даних необхідно вирішити, яке із джерел здатне на цей запит відповісти. Процеси знаходження релевантних джерел і генерування виконуваних підзапитів і складають сутність планування запитів. Планування запитів побудоване на інформації, отриманій із Сервіс-дескриптора. Нехай $\boldsymbol{R} = \{(\boldsymbol{d_1}, \boldsymbol{C_1}), \ldots, (\boldsymbol{d_n}, \boldsymbol{C_n})\}$, тут $\boldsymbol{d_1}..\boldsymbol{d_n}$ - множина джерел даних і $\boldsymbol{C_1}..\boldsymbol{C_n}$ будуть їх вмістом, де $\boldsymbol{C_i} = \{(\boldsymbol{p_{i,1}}, \boldsymbol{r_{i,1}})..(\boldsymbol{p_{i,m}}, \boldsymbol{r_{i,m}})\}$.

Також на вибір доступних джерел даних впливає можлива наявність специфікованих користувачем додаткових обмежень при інкрементарному генеруванні запиту – на яких саме джерелах даних необхідно виконувати запит.

Вибір джерела даних. Запит на мові SPARQL містить не менше одного фільтрованих базових графових шаблонів (FBGP), кожний із яких містить шаблони триплетів. Планування запиту здійснюється окремо для кожного FBGP. Алгоритм пошуку релевантних джерел даних для запиту базується на простому порів-

нянні на відповідність шаблону триплета з доступними шаблонами триплетів в джерелах даних. Відповідність визначається рівністю предиката в шаблоні триплета предикату в джерелі даних, а також відповідністю обмежень на суб'єкт і об'єкт.

Нехай BGP становить множину шаблонів триплетів в FBGP. Результатом вибору джерела даних буде множина джерел даних D_j для кожного шаблону триплетів $t_j = (s_j, p_j, o_j) \in BGP$, де $D_j = \{d | (d, C) \in R \wedge \exists (p_j, r) \in C : r(s_j, o_j) = true\}$

Побудова підзапитів. Результати вибору джерел використовуються для побудови підзапитів, на які кожне джерело може дати відповідь. Підзапити повинні складатися з одного FBGB на одне джерело даних. Подамо підзапит як триплет (T, C, d), де T – множина шаблонів триплетів, C – множина обмежень значень, d – джерело даних, яке може відповісти на підзапит.

Оптимізація запитів. За результатами планування запиту план виконання запиту складається з декількох підзапитів. Задача оптимізатора запиту полягає в тому, щоб побудувати виконуваний і ефективний план виконання запиту, який враховує обмеження на шаблони доступу. Для побудови таких запитів використовується логічна і фізична оптимізація запиту.

Логічна оптимізація. Ця оптимізація запиту використовує еквівалентність виразів запиту для перетворення логічного плану запиту в еквівалентний план, який буде виконаний швидше або з меншими витратами. Рішення, запропоноване D2RQ, здійснює логічну оптимізацію двома шляхами. Спочатку, використовуються правила, описані у праці [21] для переписування оригінального запиту перед плануванням запиту таким чином, щоб об'єднати BGP там, де це можливо і де це можливо змінні заміняються константами із виразів з фільтрами.

Потім можливі обмеження значень переміщуються в підзапити для зменшення кількості проміжних результатів настільки рано, наскільки це можливо. нехай $\mathrm{Q} = (\mathrm{T}, \mathrm{C}, \mathrm{d})$ буде підзапитом і $\mathrm{FBGP} = (\mathrm{T}', \mathrm{C}')$ – фільтрований базовий графовий шаблон. Обмеження значень C' може бути переміщене в під запит, якщо всі змінні цього обмеження також використовуються в шаблонах триплета в підзапиті. Фільтри, які містять змінні із більш ніж одного підзапиту і які не можуть бути розділені з використанням обмеженої множини правил, повинні виконуватися локально в системі федеративних запитів.

Фізична оптимізація. Ця оптимізація запиту призначена для пошуку «кращого» плану виконання запиту серед всіх доступних планів і використовує модель вартості для порівняння різних планів. Вартість запиту визначається на основі інформації щодо кількості доступних в кожному джерелі даних.

Виконання запиту. На цьому етапі можна широко застосовувати багатопоточність: скільки буде підзапитів до різних джерел даних, стільки буде виділено багатопоточних процесів. Розпаралелювання забезпечується технологіями Java Multithreading.

Об'єднання результатів запиту. Тут можна використовувати відомі результати. Дещо окремим питанням залишається реалізація *Nested Loop Join*.

Але навіть виконання цього етапу користувачем власними засобами буде виправданим, оскільки доцільність об'єднання результатів запиту не завжди легко визначити, оскільки дані дуже різнорідні.

Висновки

Запропоновано підхід до інтеграції інформаційних ресурсів, який можна застосовувати для взаємодії СЦД. Цей підхід дозволяє структурувати джерела даних і встановлювати зв'язки між ними, не вносячи змін до самих даних, структур їх зберігання та механізми супроводу. Запропонована багаторівнева архітектура системи інформаційного обслуговування користувачів за їх запитами забезпечує доступ до різнорідних джерел даних і представлення результатів у зручному вигляді. Інформація про джерела даних зберігається в онтологічних специфікаціях, з їх же допомогою описано зв'язки між властивостями джерел даних.

Рішення базується на інтеграції параметрів джерел даних з семантикою предметної області і є легко розширюваним шляхом додавання нових семантичних специфікацій і параметрів підключення джерел даних. Розроблені специфікації, що описують семантику джерел даних, можуть бути застосовані для розв'язання й інших наукових задач.

Перелік посилань

1. M. Zgurovsky, A. Gvishiani, K. Yefremov, A. Pasichny. Integration of the Ukrainian science into the world data system // Cybernetics and Systems Analysis: Volume 46, Issue 2 (2010). – P. 211.

2. Теленик С.Ф. Логічний підхід до інтеграції програмних застосувань підтримки міждисциплінарних наукових досліджень / С.Ф.Теленик, О.А.Амонс, К.В.Єфремов, В.Т.Лиско // Наукові вісті НТУУ «КПІ». – 2013. – № 5. – С. 53-72.
3. Guide To The World Data Center System - General Principles, World Data Centers, Data Services [Електронний ресурс]. Режим доступу: http://www.wdc.rl.ac.uk/wdc/guide/wdcguide.pdf.
4. Kossmann D. The state of the art in distributed query processing. – New York: ACM Computing Surveys vol.32, «ACM», 2000. – P. 422-469.
5. Тузовский А.Ф. Интеграция баз данных на основе онтологий [Електронний ресурс]. Режим доступу: ontology.ipi.ac.ru/files/d/dc/DBIntegration.ppt — Назва з екрану.
6. Имхофф К. Аналитические решения: понимание трех составляющих интеграции - EAI, EII, ETL (перевод с английского) [Електронний ресурс]. Режим доступу: http://citforum.edunet.kz/consulting/BI/integration/.
7. Wu T. EII – ETL – EAI - What, Why, and How! [Електронний ресурс]. Режим доступу: https://www6.software.ibm.com/developerworks/- tw/events/20051028/db2_6b.pdf.
8. Calvanese D., De Giacomo G., Lembo D., Lenzerini M., Rosati R. Data Management in Peer-to-Peer Data Integration Systems [Електронний ресурс]. Режим доступу: http://www.dis.uniroma1.it/~degiacom /papers/2006/calv-etal-GDM-book-06.pdf.
9. Кузнецов С.Д. Основы современных баз данных [Електронний ресурс]. Режим доступу: http://www.citforum.ru/database/osbd/- contents.shtml.
10. Интеграция данных: синтаксис и семантика / Л.Черняк // Открытые системы. – 2009. –№10. [Електронний ресурс]. Режим доступу: http://www.osp.ru/os/2009/10/ 11170978/.
11. Розробка онтології матеріалознавства засобами Protégé-OWL / Д.Г. Досин, Р.Р. Даревич, Н.В. Шкутяк // Штучний інтелект. – 2008. – № 3. – с. 70-77.
12. Д.И. Муромцев. Онтологический инжиниринг знаний в системе Protégé. – СПб: СПбГУ ИТМО, 2007. – 62 с.
13. Шаповалова С.И., Ефремов К.В., Глуханик А.И. Организация интегрированного доступа к информационным ресурсам [Текст] / С.И. Шаповалова, К.В. Ефремов, А.И. Глуханик / – Сборник трудов конференции ИАИ-2011 – 2011. – С.102-108.
14. Пол Спенсер. XML. Проектирование и реализация. М.: Лори. – 2001.– 510 с.
15. OWL Web Ontology Language W3C Recommendation [Електронний ресурс]. Режим доступу: http://www.w3.org/TR/owl-features/.
16. I. Horrocks et al. From SHIQ and RDF to OWL: the making of a Web Ontology Language / Web Semantics: Science, Services and Agents on the World Wide Web 1 (2003), p. 7–26.
17. RDF Primer W3C Recommendation [Електронний ресурс]. Режим доступу: http:// www.w3.org/TR/rdf-primer/.
18. Жигалов В. А. Технология построения естественно-языковых интерфейсов к структурированным источникам данных – М.: Диссертация на соискание ученой степени кандидата технических наук, 2000. – 172 с.
19. SPARQL Query Language for RDF [Електронний ресурс]. Режим доступу: http://www.w3.org/TR/rdf-sparql-query/.
20. ARQ - A SPARQL Processor for Jena [Електронний ресурс]. Режим доступу: http://jena.sourceforge.net/ARQ/.
21. Quilitz B., Leser U. Querying Distributed RDF Data Sources with SPARQL – Berlin: Lecture Notes in Computer Science vol. 5021, «Springer Berlin Heidelberg», 2008., – с. 524-538.
22. Langegger A., Wöß W., Blöchl M. SemWIQ – Semantic Web Integrator and Query Engine, – München: Informatik 2008 Beherrschbare Systeme dank Informatik, 2008. – с. 718-722.
23. Kikuchi S., Sachdeva S., Bhalla S. и др. Adaptive Integration of Distributed Semantic Web Data – Berlin: Databases in Networked Information Systems Volume: 5999, «Springer Berlin Heidelberg», 2010. с. 174-193.
24. A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt. FedX: A Federation Layer for Distributed Query Processing on Linked Open Data. In ESWC Poster and Demo Session Proceedings. Springer, 2011.
25. http://www.pangaea.de/
26. http://www.esimo.ru/
27. Franklin M. Fpom Database to Dataspace: A New Abstraction for Information Management / M.Franklin, A.Halevy, D.Maier // ACM SIGMOD. – 2005. – December (Record 34, no. 4). Режим доступу: http://portal.acm.org/wdc/citation.cfm7id-1107502.
28. Шаховська Н.Б. Методи опрацювання консолідованих даних за допомогою просторів даних / Н.Б.Шаховська // Проблеми програмування. – 2011. – № 4. – С. 72-84.
29. The D2RQ Plattform: Accessing Relational Databases as Virtual RDF Graphs [Електронний ресурс]. Режим доступу: http://d2rq.org/.
30. Jena Toolkit For Semantic Web Applications [Електронний ресурс]. Режим доступу: http://jena.apache.org/.
31. Evren Sirin, Bijan Parsia and oth. Pellet: A practical OWL-DL reasoner — Web Semantics: Science, Services and Agents on the World Wide Web, Volume 5, Issue 2, June, 2007, P. 51-53.
32. Бездушный А.А. Математическая модель системы интеграции данных на основе онтологий // Вестн. НГУ: Сер. Информационные технологии. Новосибирск, 2008. Т. 6. Вып. 2. с. 15–40.