УДК 004.08

РОЗРОБКА ЗАГАЛЬНОЇ АПАРАТНО-ПРОГРАМНОЇ АРХІТЕКТУРИ РОЗПОДІЛЕНОЇ ВЕРСІЇ ОНТОЛОГІЧНОГО ПОРТАЛУ

Рябова Н.В., Шевченко О.Ю., Білоіваненко М.В., Головянко М.В., Волошина Н.О., Шубкіна О.В.
Харківський національний університет радіоелектроніки

Розглянуто розробку та впровадження розподіленої версії онтологічного порталу менеджменту освітніх та наукових ресурсів МОН України. Наведено обґрунтування апаратно-програмної архітектури порталу. Проведено аналіз сучасних інтелектуальних інформаційних технологій та програмних засобів для проектування, створення і тестування онтологічних інформаційних систем, оцінку ефективності моделей архітектури онтологічного порталу з розподіленням на рівні баз даних та на рівні онтологій.

Ключові слова: *семантичні технології, онтологічний портал, архітектура, розподілена інформаційна система, кластеризація.*

Вступ

Пропонована розробка є продовженням попередніх проектів науковців кафедри штучного інтелекту Харківського національного університету радіоелектроніки, спрямованих на розробку і впровадження онтологічного порталу менеджменту та оцінки національних ресурсів України в галузі освіти та науки, і має за мету вдосконалення загальної програмно-апаратної архітектури порталу на принципах розподіленої обробки інформації та знань, яка повинна забезпечувати організацію доступу до окремих модулів, їх взаємозв'язок, пошук та видобування знань щодо інформаційних освітніх та наукових ресурсів з метою вирішення конкретних задач МОН України (насамперед, підтримки прийняття рішень щодо акредитації та ліцензування ВНЗ). Пропонована архітектура дозволить збільшити обсяг інформації, що може зберігатися в онтологічному порталі та значно підвищити швидкість його роботи.

Результатом упровадження порталу стане можливість акумуляції освітніх ресурсів в одній розподіленій Web-орієнтованій системі, автоматизація їх обробки та автоматичний розрахунок чисельних показників, які відображають динаміку здійснення освітньо-наукової діяльності кафедр і ВНЗ та відповідність цих показників нормативним документам МОН України. Такий підхід надасть можливість у режимі реального часу отримувати актуальну, несуперечливу інформацію про освітні ресурси, автоматизувати формування звітної документації. Використання розподіленого механізму функціонування порталу надасть можливість підвищення швидкості роботи системи за рахунок використання додаткових серверів ОБЗ (де зберігається вся інформація онтологічного порталу). Розробка розподіленої архітектури дозволить рівномірно розподілити навантаження на онтологічну базу між виділеними серверами (у разі великих об'ємів інформації, що буде зберігатися в цій базі).

У даній роботі використовуються результати, отримані авторами під час виконання таких попередніх проектів: «Онтологічний портал для менеджменту та оцінки національних ресурсів України в галузі освіти і науки», проект ДФФД МОН №Ф15/456-2007 (2007р.); «Розробка Web-орієнтованої системи для підтримки процедур акредитації та ліцензування вищих навчальних закладів України», НДР №219, ДР № 0107U001569 (2007-2008 р.р.); «Новітня інформаційна технологія забезпечення прозорості акредитації університетів», проект TEMPUS SM_SCM-T020B06-2006 (UA) (2007-2008 р.р.). Адреса діючого прототипу порталу http://ailab.kture.kharkov.ua/. На цей час функціональність порталу обмежена, що пов'язано з перебудовою його архітектури.

Аналіз проблемної області та постановка задач дослідження

Завдяки широкому поширенню Інтернет-технологій сучасний Web фактично перетворився в універсальний засіб доступу користувачів до інформації, розподіленої в гетерогенному інформаційному Web-середовищі. Поширення його функціональних можливостей активно розвивається завдяки парадигмі Semantic Web та відповідних технологій, спрямованих на постійне зростання можливостей представлення Web-контенту у „машино-зрозумілому" вигляді. Зокрема, семантичні властивості Web щодо можливостей спілкування агентних програм та „розуміння" власного контенту Web-системами забезпечується завдяки використанню формальних мов дескриптивної логіки, таких як RDF та OWL, а також онтологічному підходу до представлення знань. Посилення існуючих технологій Semantic Web здійснюється за допомогою так званих семантичних технологій (Semantic Technology), які з'явилися як відповідь на потребу програмно-інформаційних засобів, спроможних до експліцитного представлення значущого шару інформації та її смислової обробки. З точки зору програмного забезпечення семантична технологія передбачає кодування та зберігання файлів значущої частини (meaning) інформації окремо від файлів даних та контенту, а також окремо від коду прикладання (application). Таким чином, семантичні технології спрямовані на роботу із смислом (meaning-centered) і включають інструментарій для авторозпізнавання основних тем (topics) та понять (концептів), видобування смислового шару інформації, категорізації, смислового пошуку тощо.

Семантична технологія є одним з необхідних елементів для підтримки інфраструктур сучасного покоління інформаційних систем. Зокрема, великі організації, що мають розгалужену побудову, мають потребу в обробці складної Web-орієнтованої інформації, що постійно змінюється та є географічно розподіленою. Організація масштабних інформаційних систем згідно з концепцією розподіленої архітектури є надзвичайно актуальним питанням.

Одним з розв'язань проблеми розподілення інформації організації є побудова окремої інформаційної системи для кожної гілки організації. З розвитком семантичних технологій, що застосовуються для інжинірингу інформаційних систем, підтримка знань інформаційної системи, які подаються за допомогою онтологій, здійснюється локальними гілками організації спільно. І хоча онтології, як правило, зберігають локальну інформацію конкретного відділу організації та розміщаються в різних точках фізичного простору згідно з побудовою самої організації, необхідним є зв'язати їх між собою для глобального використання інформації, яка в них зберігається.

Головними вимогами та характеристиками онтологій в розподілених інформаційних системах є:

- мережевість. Масиви даних в різних онтологіях мають бути взаємоповязаними між собою;
- динамічність. Онтології можуть зростати в обсягах та змінюватись з часом. Тому необхідно розробити механізм динамічної підтримки онтологічних даних із моніторингом та розповсюдженням змін, що відбулися;
- розподілення. Онтології є розподіленними, що викликає проблеми, пов'язані із автономним менеджментом;
- виведення знань. Онтології, які зазвичай містять термінологічні дані та твердження, потребують підтримки процесу ефективного виведення нових знань.

Рекомендований Консорціумом W3 стандарт OWL – мова Web онтологій – надає засоби для подання та пов'язування онтологій у Web просторі в форматі, що є «зрозумілим» машиною. Однак мова Web онтологій OWL надає обмежену підтримку модульних онтологій [1]. Розроблені на сьогоднішній день технології мають низку складностей із підтримкою онтологічних даних:

- традиційно онтології, побудовані за допомогою дескриптивних логік або на основі фреймових систем, орієнтовані на централізовані онтології. Крім того, більшість

онтологічних систем менеджменту знань не підтримують обробку чисельних розподілених онтологічних об'єктів;
- в ситуації, якщо в одному з декількох пов'язаних між собою онтологічних модулів відбуваються зміни, інші пов'язані онтологічні модулі мають також обновлюватись таким чином, щоб відповідати цим змінам;
- на сьогодні відсутній підхід для підтримки розподілених онтологій, де окремі онтологічні модулі фізично розподілені та керуються в автономному режимі;
- деякі машини виведення підтримують локальне виведення за TBox-компонентом бази знань (БЗ) інформаційної системи (наприклад, FaCT++ [2]) або розподілене виведення за TBox-компонентом (наприклад, DRAGO [3]), інші реалізують виведення за ABox-компонентом (наприклад, KAON2 [4]). Однак обробка одночасно TBox та ABox в розподіленому режимі для модульних онтологій є нерозвязаною задачей для машин виведення.

Для розв'язання проблеми керування інформацією, що генерується та обробляється в розподілених середовищах, необхідно розробити формалізми подання знань та відповідні інструменти. Одним зі способів організації зберігання та обробки розподілених онтологічних знань є розподілення на рівні даних, що обробляються MySQL [5].

В даній роботі проведено докладний аналіз щодо проблеми побудови архітектури розподілених інформаційних систем на підставі онтологічного підходу та обґрунтованого вибору найбільш ефективного рішення щодо розподіленої архітектури онтологічного порталу МОНУ для надійного, безпечного та ефективного менеджменту та інтеграції освітніх ресурсів України.

Особливості розподілення онтологічної бази знань

Дворівневий метод розподілення онтологічної бази знань

На основі аналізу проблемної області щодо здійснення процедур акредитації та ліцензування у ВНЗ України, яка буда здійснена у попередніх роботах, розробники дійшли висновку, що для реалізації онтологічного порталу менеджменту освітніх ресурсів України треба застосовувати розподілення системи для більш ефективного її впровадження. Одним з підходів до розподілення є розподілення на рівні бази знань онтологічної системи.

Онтологія містить два різних типа інформації: a) перелік класів онтології, перелік властивостей, інформація про взаємодію елементів цих класів (знання, що характеризують структуру онтології), в попередніх стандартах W3C цей тип знань було відокремлено в стандарт RDFS [6]; b) перелік інформації, прив'язаної до реальних об'єктів. Ця інформація повністю відповідає загальній структурі онтології, цей тип інформації в попередніх стандартах W3C було виділено як RDF [7], як наведено на рисунку 1.

Для географічно розподіленої системи важливою особливістю є можливість локальної зміни онтологічної інформації з її наступною синхронізацією із загальною онтологічною базою. У процесі аналізу нами була обрана змішана модель розподілу інформаційних систем, що складається з головного сервера, який містить загальну структуру онтологічної інформації та набір спеціальних об'єктів, які виконують роль посилань на фрагменти онтології віддалених клієнтів. Вузли клієнтів є рівноправними та мають досить високий ступінь автономності для самостійного нормального функціонування в рамках локальних інформаційних систем, які можуть експортувати та імпортувати інформацію із глобальної онтологічної системи. Загальна структура взаємодії компонентів онтологічної системи представлена на рисунку 2.

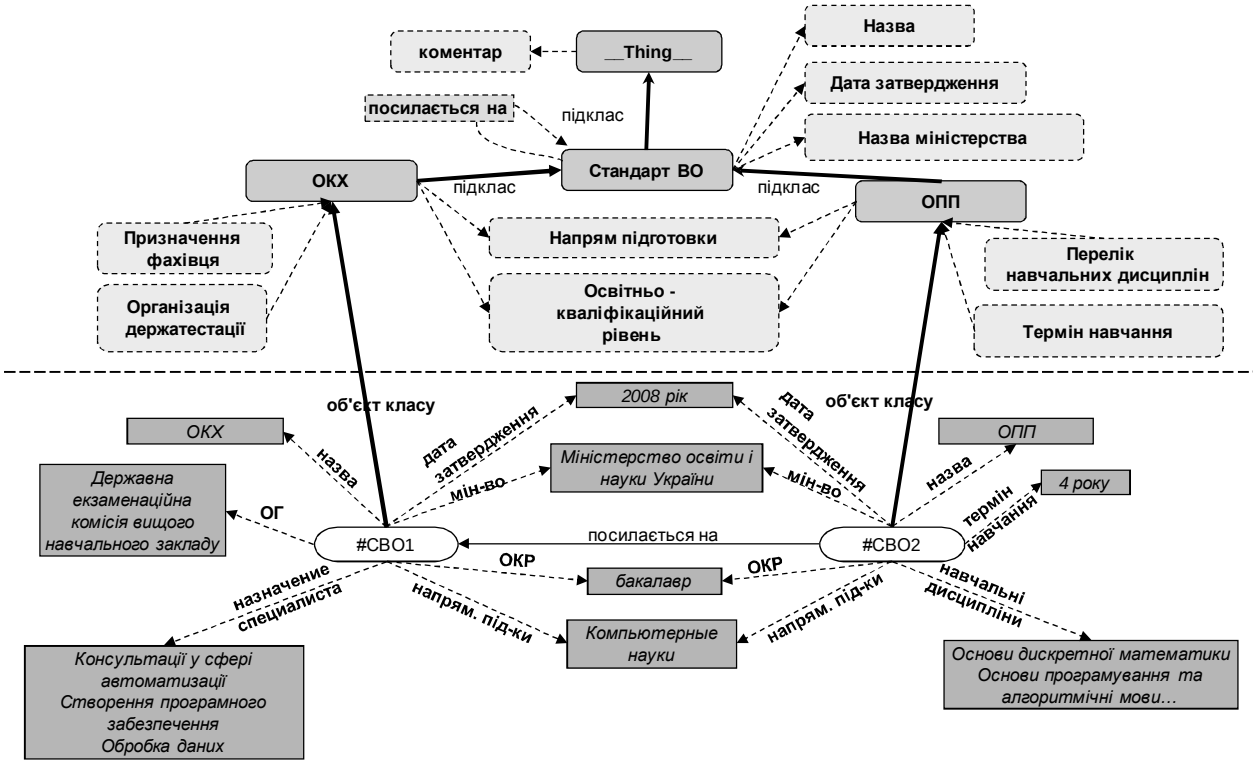

Рис. 1. Фрагмент онтології освітнього процесу (у верхній частині зображена струкрурна частина онтології, в нижній – наведено декілька прикладів опису реальних обєктів)

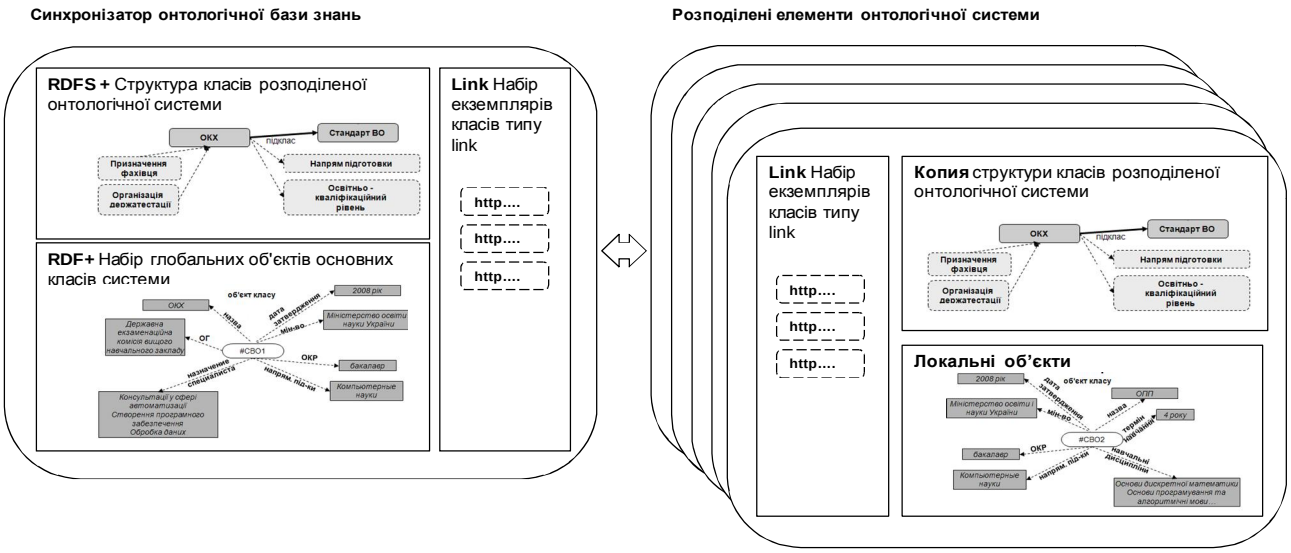

Рис. 2. Принцип побудови розподіленої онтології, яка основана на функціональному розподілі онтологічної бази знань

Глобальні інформаційні системи, як правило, складаються з компонентів, які можуть перебувати на досить великій відстані один від одного, і, навіть, якщо ці компоненти з'єднані через високошвидкісні канали, швидкість їхньої взаємодії значно знижується [8]. Тому обираючи структуру онтологічної бази, ми керувалися наступними вимогами: необхідно забезпечувати можливість роботи не тільки в рамках загальної онтологічної бази знань, але й у режимі локальної системи, у якій фрагмент онтології, що знаходиться на віддаленому клієнті, представляється як повністю незалежна онтологічна структура та може автономно функціонувати. Ці два режими можуть бути сполучені та розділятися тільки характером запиту до клієнта онтологічної бази.

Для підтримки даної функціональності необхідно підтримувати правильну структуру онтологічної бази. Як було показано на рисунку 1, онтологія складається з декількох різних

типів інформації, для підтримки нормальної роботи системи необхідно дублювати на всі клієнти онтологічної бази ту частину онтології, що відповідає за її структуру. Для коректної роботи системи ці елементи дублюються на всі клієнти онтологічної бази знань, їхня модифікація можлива тільки у випадку, якщо ці зміни були зроблені в синхронізаторі онтологічної бази знань. При цьому на синхронізуючому елементі так само повинні міститися набори глобальних об'єктів, що відповідають за роботу онтологічної бази знань. На клієнтах залишається інформація про локальні об'єкти, це дозволить легко інтегрувати існуючі інформаційні системи організацій із загальною онтологічною системою, забезпечуючи легкий обмін даними між цими системами. Такий розподіл дуже актуальний в ситуації, коли елементи онтологічної системи перебувають у різних організаціях, які мають загальне підпорядкування.

Загальний принцип розподілення об’єктів в розподіленій онтологічній системі

Розподілення об'єктів в онтологічній структурі засновано на заміщенні ряду екземплярів класу на один посилальний об'єкт, у якому зазначений принцип розподілу та перебувають посилання на всі доступні компоненти розподіленої бази знань [8]. Всі посилальні об'єкти та допоміжні об'єкти також синхронізуються на всі клієнти розподіленої онтологічної бази. Таким чином, зберігається цілісність роботи системи та забезпечується можливість взаємодії клієнтів із загальною інформацією.

Описана вище організація онтологічної бази дозволяє в значній мірі підвищити рівень швидкодії та інтеграції онтологічної системи. Для поліпшення процесу інтеграції з існуючими програмними системами та спрощення процесу модернізації й розвитку онтологічної бази знань було обрано шаблон проектування Model-View-Controller (MVC).

Шаблон проектування MVC припускає поділ даних прикладання, користувальницького інтерфейсу та керуючої логіки на три окремих компоненти: модель, представлення та контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно. Модель (Model) надає дані предметної області та реагує на команди контролера, змінюючи свій стан. Представлення (View) відповідає за відображення даних предметної області (моделі) користувачеві, реагуючи на зміни моделі. Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. У випадку з онтологічною базою знань у якості моделі реалізовано механізм взаємодії з онтологічною системою, модуль для взаємодії виконуючої програмної системи із сервером Sezam, як контролер виступає web-сервіс для взаємодії із синхронізуючим елементом розподіленої онтологічної системи та спеціальний контролер для локального доступу до онтологічної інформаційної системи. В якості “представлення” виступає локальний web-клієнт, який використовує можливість доступу з локальної системи до фрагмента загальної онтологічної структури. Завдяки подібній організації з'являється можливість легко розвивати та модифікувати інформаційну систему.

Організація взаємодії між компонентами розподіленої системи через використання Web-сервісів

Важливими питаннями організації розподіленої онтологічної системи є питання безпеки передачі інформації та питання розподілу обчислення в онтологічній системі. Для рішення цих питань на рівні контролера в моделі MVC реалізується Web-сервіс для взаємодії із синхронізуючим елементом розподіленої онтологічної системи.

Web-сервіс – програмна система, що ідентифікується рядком URI, а загальнодоступні інтерфейси визначені мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису за допомогою повідомлень, заснованих на XML, і переданих за допомогою інтернет-протоколів. Веб-Служба є одиницею модульності при використанні сервісно-орієнтованої архітектури прикладання.

Web-сервіс забезпечує потрібний рівень безпеки переданої інформації, також забезпечує розподіл рівня доступу на рівні клієнтів розподіленої онтологічної системи. Кожна команда web-сервісу є завершеною транзакцією й у випадку обриву зв'язку гарантує

збереження цілісності бази знань. Також, Web-сервіс виконує функції синхронізації фрагмента онтології, відповідального за структуру онтологічної бази. У кожний з елементів розподіленої онтологічної системи (синхронізуючий елемент або клієнт онтологічної бази) одночасно буде убудована клієнтська частина Web-сервісу та сам Web-сервіс. Якщо для виконання дії необхідно активізувати інший елемент розподіленої системи, то використовується клієнт для відправлення виклику та одержання відповіді від вилученого компонента. Якщо відбувається очікування виклику від зовнішніх систем, то в такому випадку використовується Web-сервіс. Організація взаємодії клієнта та сервісу між клієнтом і Web-сервісом здійснюється за допомогою XML - подібної мови відповідного протоколу XML-RPC. У випадку, якщо виконується запит на передачу фрагментів онтології, активізується механізм серіалізації об'єктів, за допомогою чого здійснюється передача відсутніх об'єктів на синхронізуючий елемент.

Застосування кластеризації серверу баз даних MySQL для розподілення даних

Організація збереження знань в онтологічних системах

Існує декілька підходів до організації збереження знань в розподілених онтологічних сховищах.

Підхід із використанням множини онтологій (рис. 3) застосовує декілька онтологій, кожна з яких представляє окреме джерело даних. Між онтологіями встановлюються відповідні відношення. Запити до інтегрованих даних виконуються на локальних онтологіях, а координація та розподіл запитів відбувається за допомогою встановлених відношень. У такій архітектурі немає потреби в єдиній інтегрованій онтології, крім того, як правило, зміни в локальних онтологіях не призводять до зміни відношень між онтологіями.

Підхід із використанням глобальної онтології застосовує інтегровану онтологію, що описує дані зі всіх розподілених джерел даних (рис. 4). Всі запити надсилаються до цієї єдиної онтології. Такий підхід, хоча і є найбільш очевидним способом організації розподілення, вимагає додаткових ресурсів: експерта із предметної галузі, який розуміє семантику всіх джерел даних для визначення глобальної онтології. В якості джерел можуть виступати окремі спеціалізовані онтології [9].

Гібридний онтологічний підхід (рис. 5) намагається розв’язати недоліки двох попередніх підходів. Дані в кожному джерелі подані локальною онтологією, а розподілений словник, що задається не онтологічним способом, будується для розподілення словників між локальними онтологіями. Головними перевагами цього підходу в порівнянні з попередніми двома є спрощення локального визначення онтологій та механізму побудови запитів до розподіленого словника.

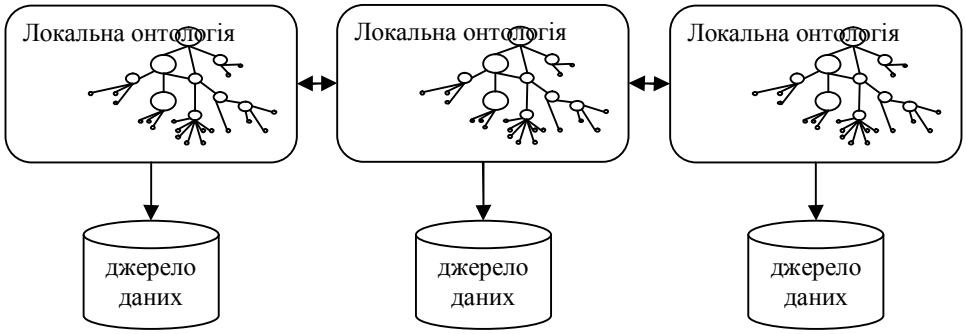

Рис. 3. Архітектура розподілення із використанням множини локальних онтології

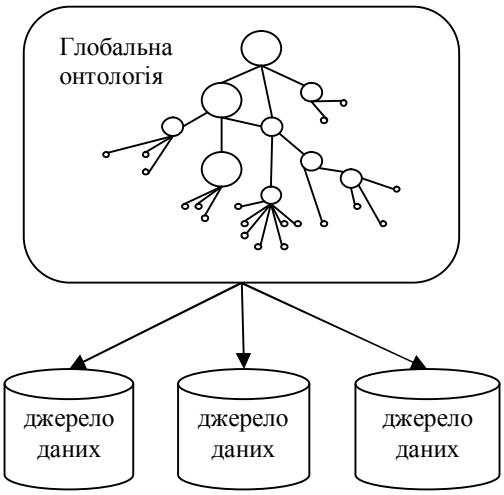

Рис. 4. Архітектура розподілення із використанням єдиної інтегрованої онтології

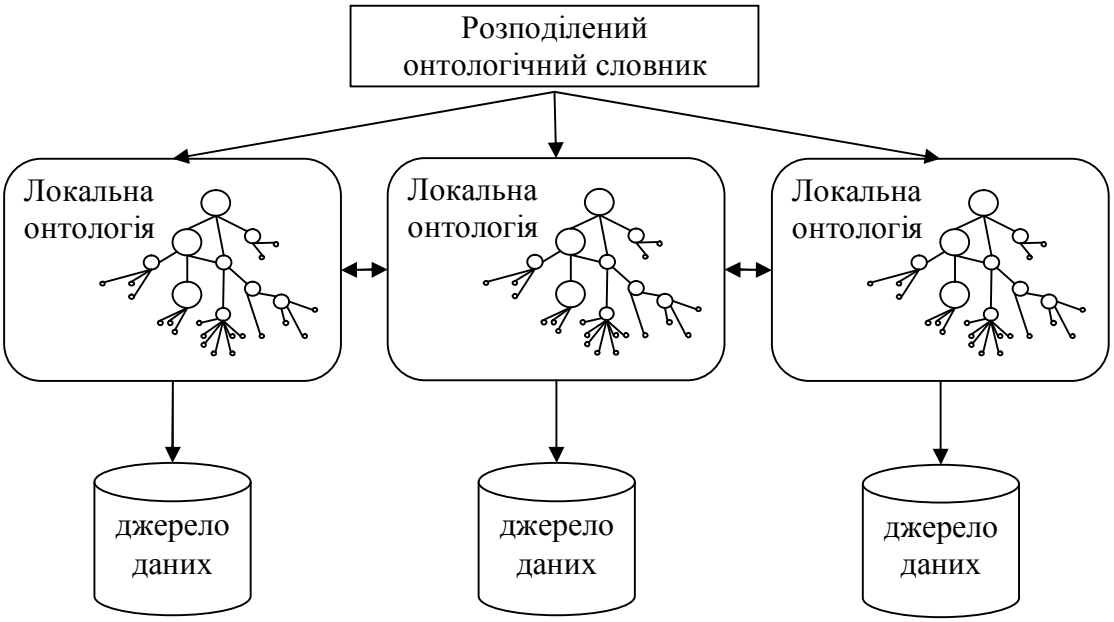

Рис. 5. Архітектура гібридного розподілення

Застосування системи Sesame для організації збереження та обробки великих обсягів інформації

Для розробки розподілених реальних систем (зокрема, Онтологічного порталу менеджменту та оцінки ресурсів в галузі освіти та науки) необхідно забезпечити можливість збереження та обробки великих обсягів RDF-даних. Одним з вирішень даної задачі є застосування технології реляційних баз даних в поєднанні з технологіями Semantic Web. Головною перевагою такого об'єднання є те, що воно надає рішення проблемі масштабування за допомогою вже розроблених механізмів.

На сьогодні Sesame може застосовувати СКБД PostgreSQL, MySQL, Oracle (9i або новіший) та SQL Server.

Для накопичення даних на рівні збереження та виведення інформації в базах даних RDBMS в Sesame застосовується динамічна схема баз даних, в якій для опису кожного нового класу та його відношень додається нова таблиця, а відношення „клас-підклас" визначає умову, що таблиця класу є підтаблицею для таблиці, що описує суперклас. Аналогічно виконується для властивостей.

Оскільки URI та літерали описуються за допомогою, як правило, довгих рядків, багато RDF-сховищ, зокрема Sesame, не зберігають рядки в таблицях триплетів, прив'язуючи рядки URI до ідентифікаторів у вигляді цілих чисел таким чином, що дані в загальному випадку нормалізуються у дві таблиці: одна таблиця триплетів, що застосовує ідентифікатори для кожного запису та інша таблиця відображення, що ставить ідентифікатори у відповідність до рядків. До цих двох загальних таблиць Sesame додає таблиці триплетів та відношень між класами та властивостями. В результаті обробки триплетів створюються:

- таблиця триплетів (addedTriples), що зберігає триплети, що було додано в явному вигляді протягом однієї транзакції;
- таблиця тверджень, які були виведені в результаті застосування одного правила виведення (allInferred);
- таблиця триплетів (allNewTriples), що зберігає триплети, додані протягом однієї транзакції;
- таблиця відношень (direct_subclassof), що зберігає всі прямі відношення типу «клас-підклас»;
- таблиця відношень (direct_subpropertyof), що зберігає всі прямі відношення типу «властивість-підвластивість»;
- таблиця предметної галузі (domain);
- таблиця (instanceof), що задає відношення «бути екземпляром класу»;
- таблиця літералів (literals);
- таблиця, що зберігає простори імен (namespaces);
- таблиця триплетів (newTriples), що зберігає нові триплети, що було додано;
- та інші.

З'єднання з MySQL може реалізовуватись двома шляхами:

1) може існувати статичний (попередньо створений) репозиторій, який конфігурується за допомогою Configure Sesame для можливості застосування MySQL БД;

2) налаштування БД може відбуватися за допомогою редагування вихідного коду прикладання.

Оскільки Sesame є незалежним від СКБД, весь СКБД-специфічний код зберігається в одному окремому рівні: рівні збереження та виведення.

Кластеризація в MySQL

У класичній базі даних MySQL дані організовані у вигляді таблиць, ці таблиці зберігаються як файли на диску серверу баз даних [5]. Кластеризація розподіляє обробку даних між декількома серверами.

Кластер серверів дозволяє об'єднати декілька фізичних серверів (вузлів), які виступають партнерами один для одного в процедурі переходу на резервний ресурс.

Існує багато причин для кластеризації бази даних та декілька різних способів кластеризації.

Вертикальне масштабування (scaling up) є традиційним методом, що застосовується у випадку, коли поточне устаткування стає застарілим. Адміністратори проводять модернізацію устаткування на рівні поточного серверу або замінюють його на більш потужний. Горизонтальне масштабування (scaling out) означає, що додається більше серверов, а устаткування залишається старим. Другий спосіб вважається більш досконалим з огляду на надійність та витрати. Однак він є і більш складним. Крім того, витрати на програмне забезпечення є суттєвим.

Основні причини кластеризації бази даних:

1) високий рівень доступності даних. Забезпечення можливості повної відмови окремих серверів всередині кластеру без будь-якої зупинки роботи користувачів БД за рахунок властивості надлишковості даних, що зберігаються в кластері.;

2) масштабування. Можливість додавання устаткування до кластеру – прозоро для користувачів БД – для підвищення ефективності. Це дозволяє запускати БД на великій

кількості бюджетних комп'ютерів, і додавати нові бюджетні комп'ютери до кластеру у разі потреби.

Крім того, кластеризація забезпечує можливість організації окремих серверів БД з розподіленими даними, вдосконаленими налаштуваннями реплікації та іншими кластерами БД, простішого менеджменту (наприклад, всі вузли в кластері можуть контролюватися з однієї машини), менших витрат та більшої надійності в цілому.

Існує два головних методи кластеризації БД: кластеризація без розподілення ресурсів (кожен сервер володіє власними дисковими ресурсами), та кластеризація із розподіленням даних з диску (декілька серверів БД зчитують інформацію з одного диску).

Для побудови кластеру необхідно:

- організувати мережу (всі вузли кластеру мають бути пов'язаними за допомогою з'єднання із мінімальною пропускною здатністю в 100 Mbps);
- визначити кількість реплік;
- визначити обсяг RAM-пам'яті;
- визначити обсяг дискового простору;
- визначити оперативну систему.

MySQL Cluster – це технологія, яка уможливлює кластеризацію баз даних, що знаходяться в пам'яті системи, організованої без розподілення ресурсів. MySQL Cluster побудований таким чином, що кожен компонент системи має свою окрему пам'ять та диск. MySQL Cluster інтегрує стандартний MySQL-сервер із вбудованою кластерованою машиною виведення NDB, що працює зі сховищем даних.

MySQL кластер складається з:

- SQL-вузлів множини комп'ютерів, що запускають MySQL сервери для отримання та відповіді на запити. SQL-вузли є спеціалізованим типом API-вузлів, які описують прикладання, що звертаються до кластеру;
- вузлів даних (strorage nodes) для зберігання даних, що містяться в кластері, та обробки запитів. Кількість вузлів даних залежить від кількості реплік та фрагментів даних;
- одного чи декількох вузлів менеджменту (MGM node) або серверів керування, що виступають в ролі центральної точки доступу для роботи із внутрішнім кластером (роллю цього типу вузлів є керування іншими вузлами всередині кластеру, виконання таких функцій, як надання конфігураційних даних, запуск та зупинка вузлів, запуск запасного збереження даних та інших);
- спеціалізованих програм доступу до даних.

Кластер MySQL інтегрує стандартний MySQL-сервер із вбудованим в пам'ять машиною виведення NDB, що характеризується такими якостями, як високий доступ та стійкість даних.

MySQL кластер може використовуватись з існуючими MySQL-прикладаннями. Такі клієнтські прикладання надсилають SQL-твердження та отримують відповідь від MySQL-серверів, що працюють як SQL-вузли кластеру таким самим чином, як окремі MySQL-сервери. Однак MySQL-клієнти, що застосовують кластер MySQL як джерело даних можуть змінюватися для отримання переваг з'єднання з різними MySQL-серверами для досягнення вирівнювання завантаження та перехват керування у випадку відмови. Схему взаємодії компонентів представлено на рисунку 6 [5].

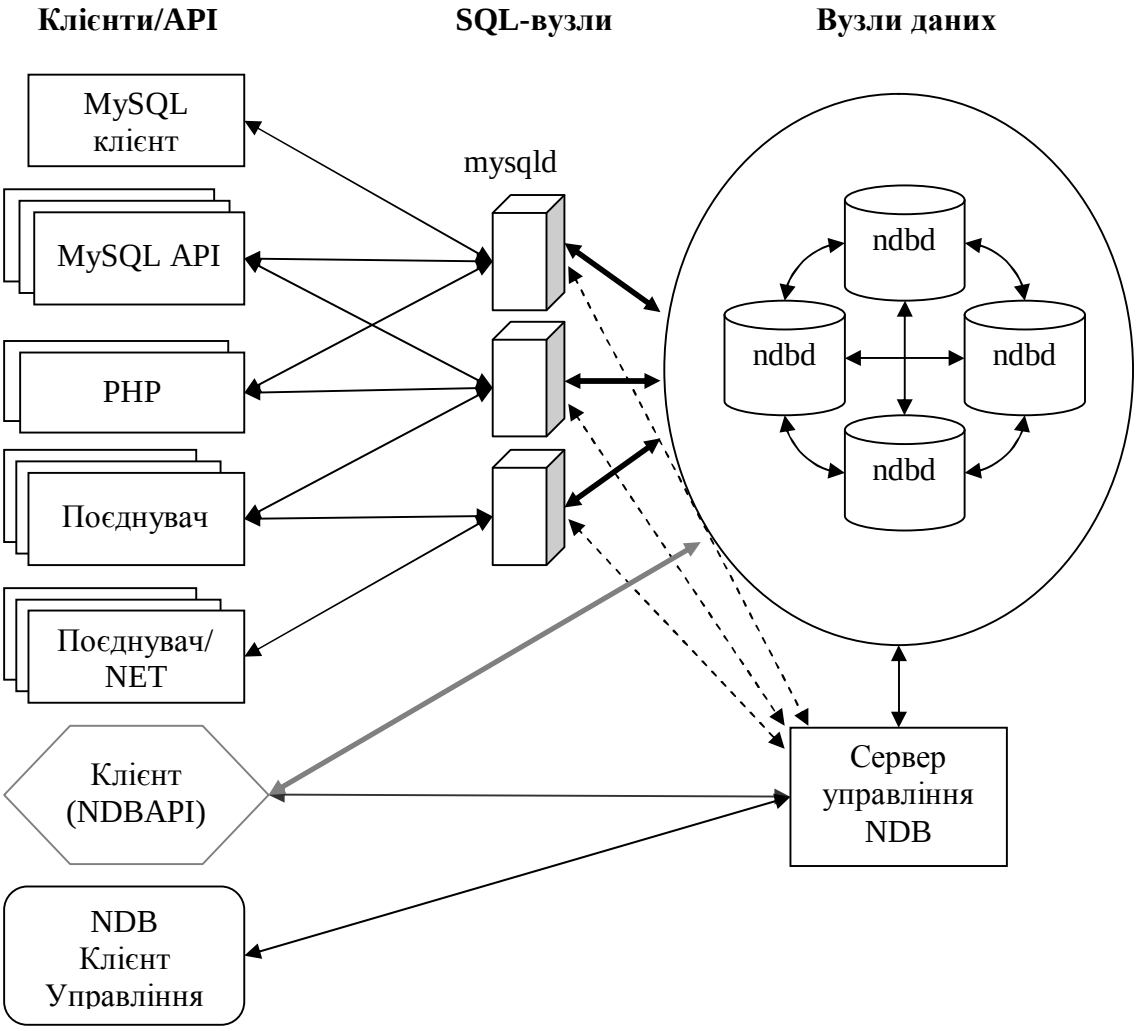

Рис. 6. Схема підключення кластеру

Реплікація – це процес копіювання даних однієї БД між іншими БД в режимі online для розв'язання задач підвищення доступності та надійності збереження даних.

Master Server – головний сервер, з якого відбувається копіювання.

Slave Server – підлеглий сервер, на який копіюються дані.

Всі зміни, що відбуваються на Master-сервері синхронізуються на Slave-сервері. Slave-сервери з певною періодичністю опитують Master-сервер на предмет змін в базі. Таким чином, створюється надлишковість даних на декількох серверах. Важливою особливістю є те, що кожного разу переносяться лише зміни, що відбулись в даних, а не всі дані. Master-сервер записує всі зміни в бінарний журнальний лог, присвоюючи кожній операції свій номер, та протоколює ротації бінарних журналів в індексному файлі. Коли Slave-сервери звертаються до головного сервера, він повідомляє номер останньої операції, яка була виконана, та отримує всі нові зміни, відраховуючи від цього номера.

MySQL-кластер має чотири різні методи отримання даних з різними експлуатаційними характеристиками: доступ через первинний ключ, доступ через унікальний ключ, доступ за впорядкованим індексом та перегляд повної таблиці.

Доступ через первинний ключ відбувається, коли в запиті застосовується первинний ключ з використанням пошуку в хеші. MySQL-сервер створює хеш первинного ключа і потім, використовуючи алгоритм розподілення даних, знає точно, які вузли даних містять потрібну інформацію та видобуває їх звідти, як вказано на рисунку 7.

Процес відбувається за одним сценарієм незалежно від кількості вузлів даних.

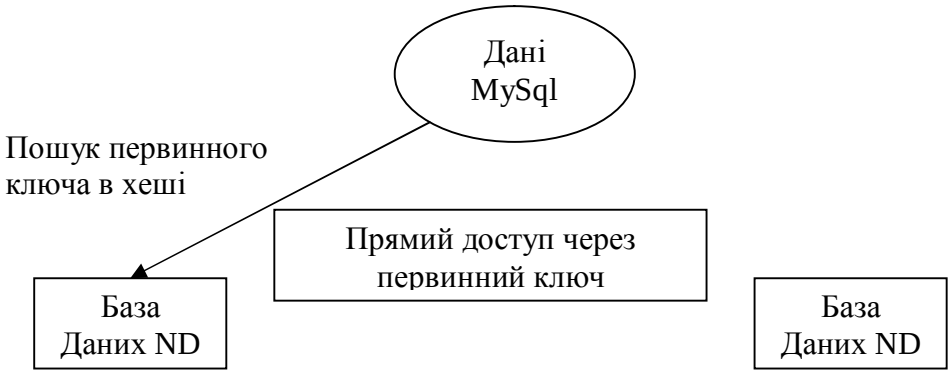

Рис. 7 . Доступ через первинний ключ

Доступ через унікальний ключ відбувається, коли запит застосовує унікальний ключ для доступу до даних. У цій ситуації відбувається пошук за хешем. Однак цей тип доступу є двокроковим процесом, як наведено на рисунку 8.

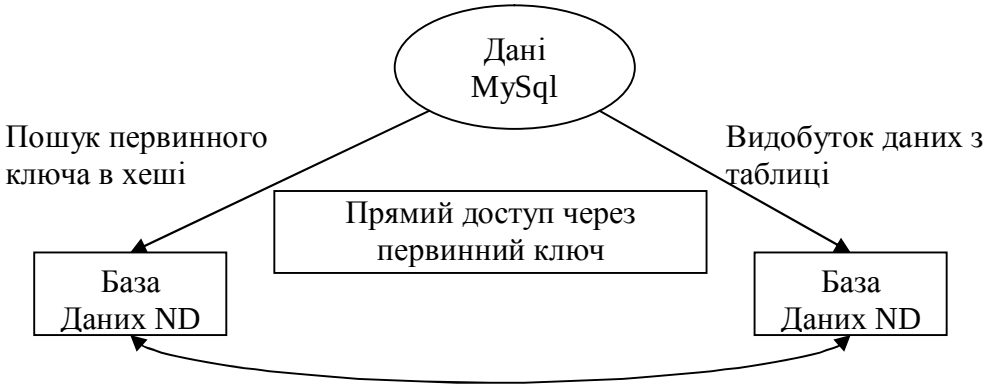

Рис. 8. Доступ через унікальний ключ

Для доступу за впорядкованим індексом MySQL-сервер виконує операцію, що має назву паралельне сканування індексу, як зображено на рисунку 9. Це означає, що необхідно змусити кожен вузол даних продивитися всі індекси, які зберігаються на ньому локально. Вузол робить це паралельно, а MySQL-сервер комбінує результати, коли вони повертаються на сервер.

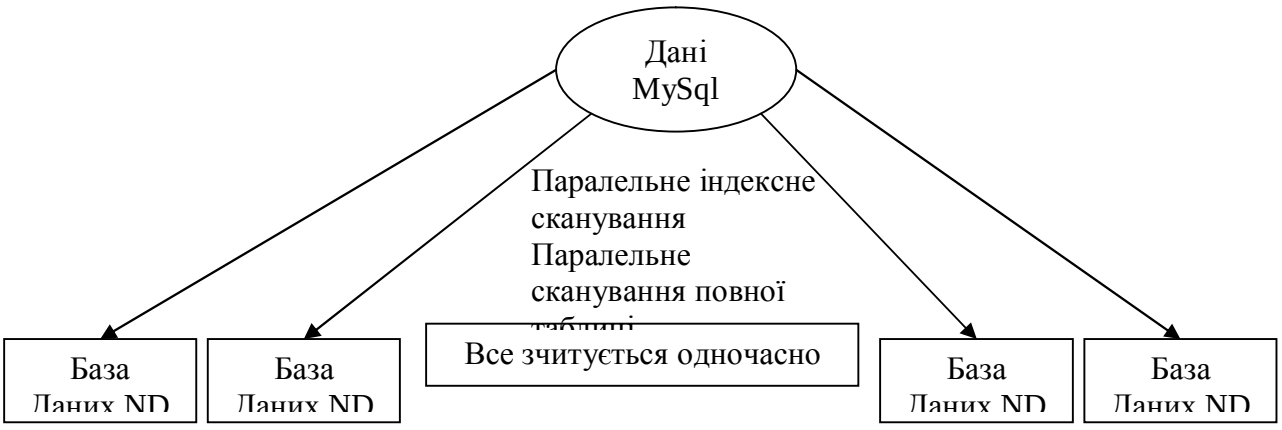

Рис. 9. Доступ за впорядкованим індексом

Останній метод – повного сканування таблиці – може відбуватися двома шляхами, в залежності від версії та налаштувань MySQL-серверу: повне сканування таблиці без умов та повне сканування таблиці з умовами.

Апаратно-програмна архітектура онтологічного порталу з розподіленням онтології на рівні даних

Переваги розподілення онтологічного порталу на рівні даних

Для розподілення онтологічного порталу менеджменту та оцінки національних освітніх ресурсів України після проведення аналізу технологій розподілення, які можливо застосовувати до онтологічної системи, було обрано модель розподілення на рівні даних.

Розподілення на рівні реалізації СКБД вирішує наступні задачі:

- розподілення даних по групах на різних серверах, що знизить навантаження на носії даних та підвищить швидкість доступу до сховищ даних;
- організація балансування навантаження системи на декількох серверах, що підвищує її ефективність роботи при підключенні декількох користувачів одночасно;
- підвищення надійності та відмовостійкість системи за рахунок дублювання даних.

При реалізації такого підходу розробка нових програмних модулів в існуючій системі не потрібна, бо кластером СКБД є готовий до використання існуючий продукт. З точки зору розподіленої архітектури онтологічного порталу існуюча система може стати звичайним сервером баз даних, тому може бути «прозоро» включена в існуючий прототип порталу за обраною схемою розподілення.

Також для реалізації розподіленої версії архітектури онтологічного порталу необхідно створити нові вимоги до адміністрування порталу в чистині підтримки кластеру СКБД та інфраструктуру серверів для реалізації кластеру.

Архітектура кластера системи керування базами даних

Для розподілення онтологічного порталу менеджменту національних освітніх ресурсів України було обрано організацію кластеру на основі СКБД MySQL версії 5.1 [5]. Типова архітектура такого кластеру представлена на рисунку 10.

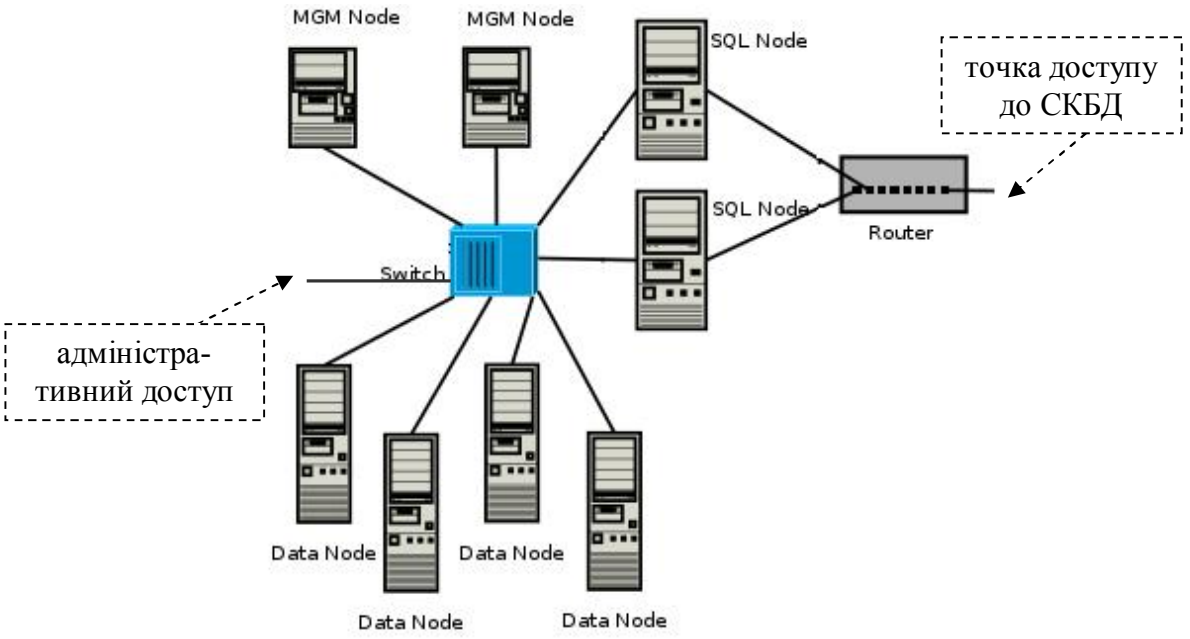

Рис. 10. Архітектура кластеру MySQL

Інфраструктура кластеру містить у собі наступні типи компонент:

1) MGM Node – вузол керування кластером;

2) SQL Nod – вузол обробки запитів, забезпечує виконання SQL-команд клієнтів (в межах онтологічного порталу – це сервери підтримки онтологій). Основна загрузка на сервер складається при обробці запитів, що потребує підвищення вимог до швидкості та кількості процесорів, а також до об’єму ОЗП;

3) Data Node – вузол даних, що забезпечує дзеркальне (повне копіювання даних) або додаткове (декілька вузлів створюють єдиний простір даних) сховище даних. Такий сервер буде навантажений при доступі до даних, що підвищує вимоги до системи керування дисками та об'ємами носіїв.

У такій архітектурі виділяються два типи активного мереженого обладнання:

1) Switch – внутрішній комутатор для зв'язку всіх серверів між собою, який повинен мати максимальну пропускну можливість, можливість підключення спеціального клієнту для адміністрування кластеру;

2) Router – маршрутизатор та балансувальник, який виконує функції обмеження доступу для зовнішніх клієнтів до кластеру та розподілення навантаження (шляхом послідовного перебирання маршрутів) на SQL-сервери; повинен мати додаткові функції маршрутизації.

Архітектура розподіленого онтологічного порталу

Після проведення аналізу існуючої системи, схем та моделей розподілення онтологічних систем та програмного забезпечення, що дає можливість створення та керування розподіленими онтологічними системами, було розроблено загальну архітектуру розподіленої версії онтологічного порталу підтримки акредитації та ліцензування національних освітніх ресурсів України. Розроблена архітектура представлена на рисунку 11 [10].

Розподілена архітектура містить у собі наступні програмні сервери: Apache Tomcat – підтримка HTTP-запитів Web-клієнтів, сервер статичних файлів зображень та документів, сервер прикладань для ядра та програмних модулів порталу; Aduna Sesame – сервер підтримки RDF-графу та онтологічної моделі даних; MySQL Cluster – набір серверів трьох типів, які реалізують кластер СКБД.

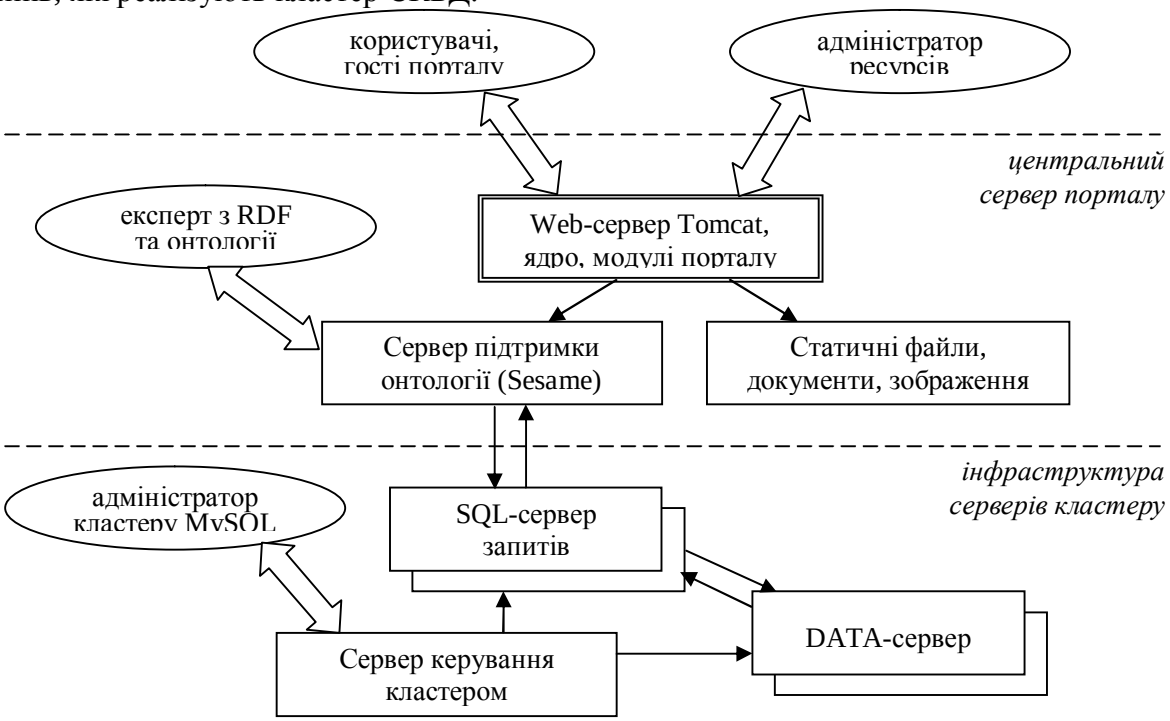

Рис. 11. Архітектура розподіленого онтологічного порталу

Структурна схема онтологічного порталу, який було розроблено на попередніх ітераціях проектування системи, представлена на рисунку 12.

У зв'язку з використанням розподілення системи за допомогою організації кластеру СКБД, онтології модулів системи винесено в загальне сховище онтологічних структур даних Sesame. При цьому модифікація програмного коду буде непотрібна, бо кластер СКБД має такий самий SQL-інтерфейс, що і автономний сервер СКБД MySQL.

Щодо змін, які повинні будуть відбутися в системі при створенні розподіленої її модифікації, потрібно організувати додаткову конфігурацію рапозиторію та стуку зберігання даних у стандартних настройках серверу Sesame.

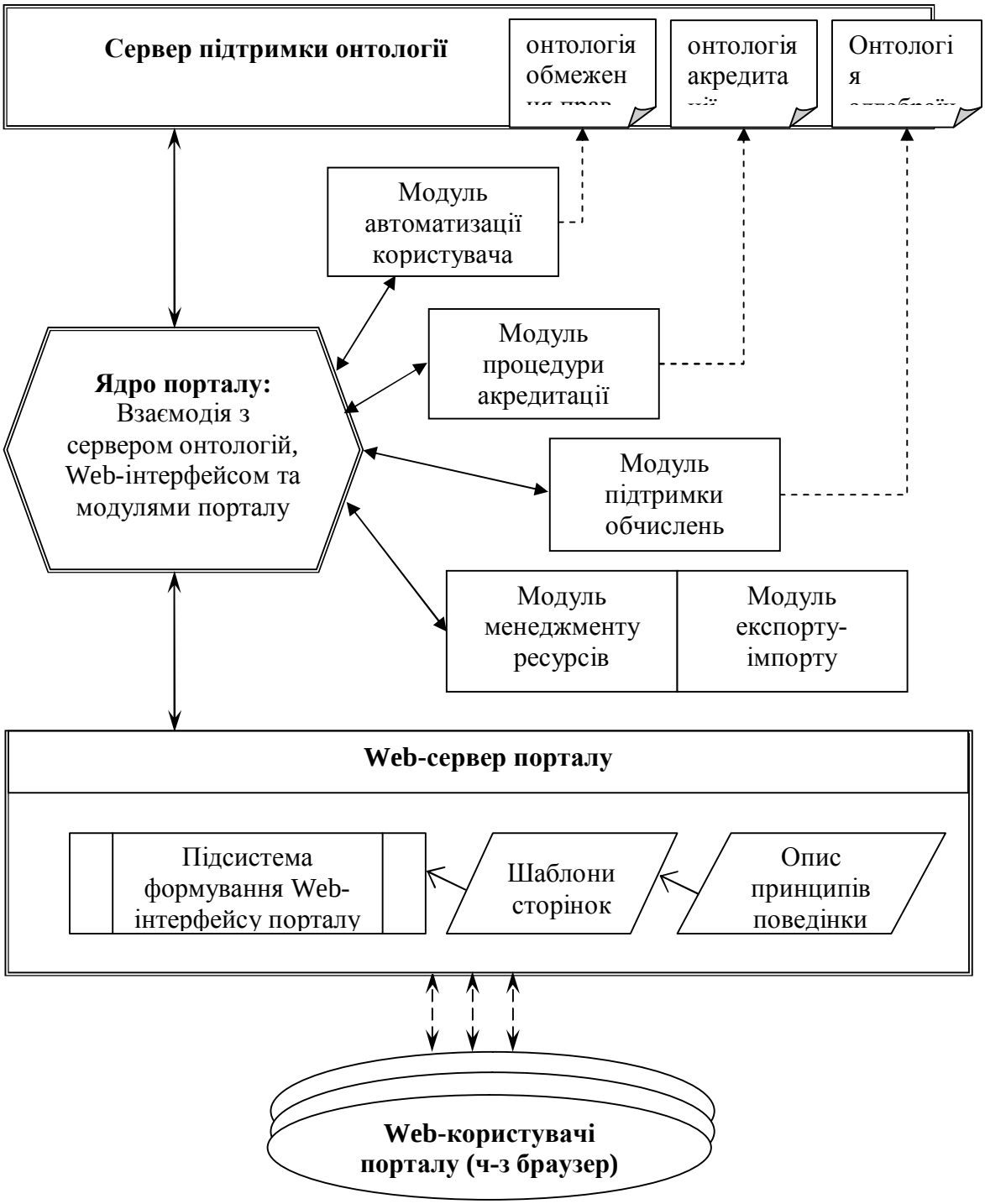

Рис. 12. Структурна схема онтологічного порталу

Висновки

У результаті виконання роботи було отримано наступні результати:

- проведено порівняльний аналіз моделей та архітектур розподілення інформаційних систем, які побудовані на основі онтологічного підходу; було розглянуто моделі розподілення, що передбачають розподілення інформаційної системи як на фізичному рівні, так і на рівні бізнес-логіки системи;
- здійснено аналіз сучасних СКБД та моделей організації розподілених систем на їх основі; в результаті була обрана СКБД MySQL як така, що повністю відповідає вимогам розподіленої системи, що розроблюється, дозволяє сформувати

ефективне представлення інформації системи за допомогою формування кластерів;

- проведено аналіз сучасних існуючих програмних середовищ, що підтримують обробку онтологічних структур віддалено; в результаті було обрано сервер SESAME; приводом для обрання цього серверу зберігання та обробки онтологічних структур стало те, що існуюча версія порталу вже побудована з його використанням, він містить гнучкі механізми обробки онтологічних структур та реалізує взаємодію з сервером MySQL на основі використання Java-машини;
- на основі аналізу моделей розподілення програмних продуктів, які необхідно використовувати для розподілення онтологічної системи, було обрано модель розподілення на рівні даних та розроблено програмно-апаратну архітектуру розподіленої версії онтологічного порталу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OWL Specification Development. [Електронний ресурс] // Режим доступу: www.w3.org/2004/OWL/
2. FaCT++ Description Logic Reasoner: System Description. [Електронний ресурс] // Режим доступу: www.comlab.ox.ac.uk/people/ian.horrocks/Publications/download/2006/TsHo06a.pdf
3. DRAGO, System description. [Електронний ресурс] // Режим доступу: http://drago.itc.it/system-description.html
4. KAON2 – Ontology Management for the Semantic Web. [Електронний ресурс] // Режим доступу: http://kaon2.semanticweb.org/
5. Офіційний сайт розробника MySQL. [Електронний ресурс] // Режим доступу: http://www.mysql.com/
6. Broekstra J., Kampman A., Harmelen F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema // International Semantic Web Conference 2002, Sardinia, Italy. [Електронний ресурс] // Режим доступу: http://www.openrdf.org/doc/papers/Sesame-ISWC2002.pdfs
7. RDF Specification Development. [Електронний ресурс] // Режим доступу: RDF www.w3.org/RDF/
8. Рябова Н.В., Волошина Н.А., Шубкина О.В., Шаламов М.А. Построение онтологической базы знаний для системы управления web-контентом // Материалы междунар. науч.-практ. конф.: «Информационные технологии и информационная безопасность в науке, технике и образовании «ИНФОТЕХ-2009». – Севастополь: Изд-во СЕВНТУ, 2009. – С. 230-233.
9. Климова М.В., Шевченко А.Ю. Метод побудови інтелектуальних систем обробки інформації та документообігу за допомогою онтологічної бази знань // Штучний інтелект.- №2. – 2009. – Донецьк. – С.91-97.
10. Воскобойникова А.А. Мультиагентный интерфейс для доступа к онтологической системе в архитектуре интеграции информационных систем. Information Science and Computing, IBS Number 12, Human Aspects of Artificial Intelligence, ITHEA, Sofia, 2009.