Розглядається розробка та впровадження розподіленої версії онтологічного порталу менеджменту освітніх та наукових ресурсів МОНМС України. Наведено обгрунтування апаратно-програмної архітектури порталу

Ключові слова: семантичні технології, онтологічний портал, архітектура

Рассмотривается разработка и внедрение распределенной версии онтологического портала менеджмента образовательных и научных ресурсов МОНМС Украины. Приведено обоснование аппаратно-программной архитектуры портала

Ключевые слова: семантические технологии, онтологический портал, архитектура

Development of the distributed version of the ontology-based portal for management of education and scientific resources of MESYS is considered. The substantiation of the hardware and software architecture of the portal

Keywords: semantic technologies, ontology-basedl portal, architecture

УДК 004.75:[004.089:002.53]

РАЗРАБОТКА ОБОБЩЕННОЙ АППАРАТНО-ПРОГРАММНОЙ АРХИТЕКТУРЫ РАСПРЕДЕЛЕННОЙ ВЕРСИИ ОНТОЛОГИЧЕСКОГО ПОРТАЛА

Н.В. Рябова
Кандидат технических наук, доцент,
исполняющая обязанности заведующего кафедры*
А.Ю. Шевченко
Кандидат технических наук, доцент*
М.В. Белоиваненко
Научный сотрудник*
М.В. Головянко
Кандидат технических наук, старший преподаватель*
Н.А. Волошина
Старший преподаватель*
О.В. Шубкина
Кандидат технических наук, асситент*
А.А. Воскобойникова
Ассистент*
*Кафедра искусственного интеллекта
Харьковский национальный университет радиоэлектроники
пр. Ленина, 16, г. Харьков, 61166
E-mail: ai@kture.kharkov.ua

Введение

Данная работа является продолжением предыдущих проектов ученых кафедры искусственного интеллекта Харьковского национального университета радиоэлектроники, направленных на разработку и внедрение онтологического портала менеджмента и оценки национальных ресурсов Украины в области образования и науки. Целью разработки является усовершенствование аппаратно-программной архитектуры портала, основанной на принципах распределения информационных ресурсов и знаний. Такой подход должен обеспечивать организацию доступа к отдельным модулям, их взаимосвязь, поиск и извлечение знаний, связанных с информационными образовательными и научными ресурсами для решения конкретных задач МОНМС Украины (прежде всего, поддержки принятия решений при проведении процедур аккредитации и лицензирования ВУЗов). Предложенная архитектура позволит увеличить объем информации, которая может храниться в онтологическом портале и значительно увеличить скорость его работы.

В результате внедрения портала будет возможна аккумуляция образовательных ресурсов в одной распределенной Web-ориентированной системе, автоматизация их обработки и автоматический расчет численных показателей, которые отображают динамику развития образовательно-научной деятельности кафедр и ВУЗов и соответствие этих показателей нормативным документам МОНМС Украины. Предлагаемый подход даст возможность в режиме реального времени получать актуальную, непротиворечивую информацию об образовательных ресурсах, автоматизировать формирование отчетной документации. Использование распределенного механизма функционирования портала даст возможность увеличения скорости работы системы за счет использования дополнительных серверов онтологической базы знаний (ОБЗ), где хранится вся информация онтологического портала. Разработка распределенной архитектуры позолит равномерно распределить нагрузку на онтологическую базу между выделенными серверами (в случае больших объемов информации, которая будет храниться в этой базе).

В данной работе используются результаты, полученные авторами при выполнении предыдущих проектов: «Онтологический портал для менеджмента и оценки национальных ресурсов Украины в области образования и науки», проект ДФФД

МОН №Ф15/456-2007 (2007г.); «Разработка Web-ориентированной системы для поддержки процедур аккредитации и лицензирования высших учебных заведений Украины», НИР №219, ДР № 0107U001569 (2007-2008г.г.); «Современная информационная технология обеспечения прозрачности аккредитации университетов», проект TEMPUS SM_SCM-T020B06-2006 (UA) (2007-2008 г.г.). Адрес действующего прототипа портала http://ailab.kture.kharkov.ua/. На сегодня функциональность портала ограничена в связи с усовершенствованием его архитектуры.

1. Анализ проблемной области и постановка задачи исследования

Благодаря широкому распространению Интернет-технологий, современный Web фактически превратился в универсальное средство доступа пользователей к информации, распределенной в гетерогенном информационном пространстве. Расширение его функциональных возможностей активно развивается благодаря парадигме Semantic Web и соответствующих технологий, направленных на постоянное увеличение возможностей представления Web-контента в «машинно-понятном» виде. Например, семантические свойства Web относительно возможностей общения агентных программ и «понимания» собственного контента Web-системами обеспечивается благодаря использованию формальных языков дескрипционных логик, таких как RDF и OWL, а также онтологическому подходу к представлению знаний. Усиление существующих технологий Semantic Web осуществляется с помощью так называемых семантических технологий (Semantic Technology), которые появились и активно развиваются в силу назревшей необходимости в программно-информационных средствах, обеспечивающих эксплицитное представление значимого слоя информации и ее смысловую обработку. С точки зрения программного обеспечения семантическая технология предусматривает кодирование и хранение файлов значимой части (meaning) информации отдельно от файлов данных и контента, а также отдельно от кода приложения (application). Таким образом, семантические технологии направлены на работу со смыслом (meaning-centered) и включают инструментарий для авторазпознавания основных тем (topics) и понятий (концептов), извлечение смыслового слоя информации, категоризацию, смысловой поиск и т.п. Семантическая технология является одним из необходимых элементов для поддержки инфраструктур современного поколения информационных систем. Например, крупные организации с разветвленной инфраструктурой испытывают необходимость в обработке сложной Web-ориентированной информации, которая постоянно изменяется и является распределенной географически. Организация масштабных информационных систем согласно концепции распределенной архитектуры является черезвычайно актуальным направлением развития современных IT-технологий.

Одним из решений проблемы распределения информации организации является построение отдельной информационной системы для каждой ветви организации. С развитием семантических технологий, которые применяются для инжиниринга информационных систем, поддержка знаний информационной системы, формализуемых и представляемых с помощью онтологий, осуществляется локальными ветвями организации совместно. Такие онтологии, как правило, хранят локальную информацию конкретного отдела организации и размещаются в разных точках физического пространства согласно структуре самой организации, потому необходимо связать их между собой для глобального использования хранимой в них информации. Главными требованиями и характеристиками онтологий в распределенных информационных системах являются:

- наличие сетевой связи данных. Массивы данных в разных онтологиях должны быть взаимосвязанными между собой;
- динамичность. Онтологии могут расти в объемах и изменяться со временем. Поэтому необходимо разработать механизм динамической поддержки онтологических данных с мониторингом и распространением изменений, которые произошли;
- распределение. Онтологии являются распределенными, что вызывает проблемы, которые связаны с автономным менеджментом;
- вывод знаний. Онтологии, которые обычно содержат терминологические данные и утверждения, требуют поддержки процесса эффективного вывода новых знаний;

Рекомендованный Консорциумом W3 стандарт OWL – язык Web онтологий – дает средства для представления и связывания онтологий в Web-пространстве в формате, который является «понятным» машине. Однако, язык Web онтологий OWL предоставляет ограниченную поддержку модульных онтологий [1]. Разработанные на сегодня технологии имеют ряд сложностей с поддержкой онтологических данных:

- традиционно онтологии, построенные с помощью дескрипционных логик или на основе фреймовых систем, ориентированы на централизованные онтологии. Кроме того, большинство онтологических систем менеджмента знаний не поддерживают обработку множественных распределенных онтологических объектов;
- в ситуации, если в одном из нескольких связанных между собой онтологических модулей происходят изменения, другие связанные онтологические модули должны также обновляться таким образом, чтобы соответствовать этим изменениям;
- на сегодня не существует подхода для поддержки распределенных онтологий, где отдельные онтологические модули физически распределены и управляются в автономном режиме;
- некоторые машины вывода поддерживают локальный вывод по TBox-компоненту базы знаний (БЗ) информационной системы (например, FaCT++ [2]) или распределенный вывод по TBox-компоненту (например, DRAGO [3]), другие реализуют вывод по ABox-компоненту (например, KAON2 [4]). Однако, обработка одновременно TBox и ABox в распределенном режиме для модульных онтологий является нерешенной задачей для машин вывода.

Для решения проблемы управления информацией, которая генегируется и обрабатывается в рас-

пределенных средах, необходимо разработать формализмы представления знаний и соответствующие инструменты. Одним из способов организации хранения и обработки распределенных онтологических знаний является распределение на уровне данных, которые обрабатываются MySQL [5].

В данной работе проведен подробный анализ проблемы построения архитектуры распределенных информационных систем на основе онтологического подхода и обоснованного выбора наиболее эффективного решения для распределенной архитектуры онтологического портала МОНМС Украины для надежного, безопасного и эффективного менеджмента и интеграции образовательных ресурсов Украины.

2. Особенности распределения онтологической базы знаний

2.1. Двухуровневый метод распределения онтологической базы знаний

На основании анализа предметной области по проведению процедур аккредитации и лицензирования в ВУЗах Украины, разработчики пришли к выводу, что для создания онтологического портала менеджмента образовательных ресурсов Украины необходимо использовать распределение онтологической системы на уровне базы знаний.

Онтология содержит два различных типа информации: а) перечень классов онтологии, перечень свойств, информацию о взаимодействии элементов этих классов (знания, которые характеризуют структуру онтологии), в предыдущих стандартах W3C тот тип знаний был выделен в стандарт RDFS [6]; б) информацию, привязанную к реальным объектам. Такая информация полностью соответствует общей структуре онтологии, этот тип информации в предыдущих стандартах W3C был выделен в стандарт RDF [7] (рис. 1).

Для географически распределенной системы важной особенностью является возможность локального изменения онтологической информации с последующей синхронизацией с обобщенной онтологичной базой. В процессе анализа мы выбрали смешанную модель распределения информационных систем, которая состоит из главного сервера, который содержит общую структуру онтологической информации и набор специальных объектов, которые выполняют роль ссылок на объекты онтологии отдаленных клиентов. Узлы клиентов являются равноправными и имеют достаточно высокий уровень автономности для самостоятельного нормального функционирования в рамках локальных информационных систем, которые могут экспортировать и импортировать информацию из глобальной онтологической системы. Обобщенная структура взаимодействия компонентов онтологической системы представлена на рис. 2.

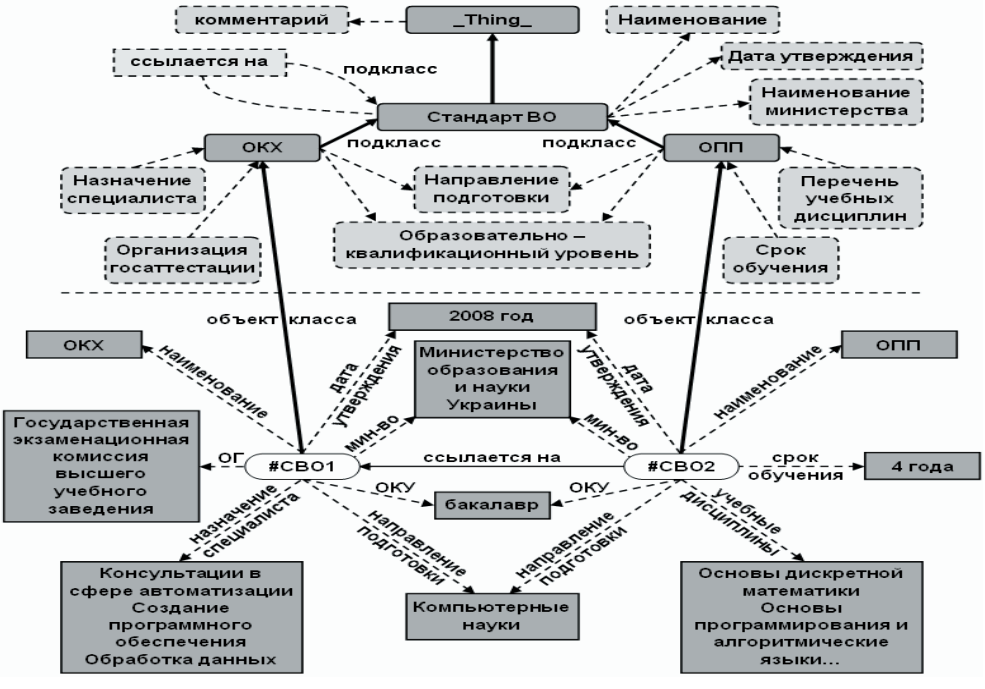

Рис. 1. Фрагмент онтологии образовательного процесса (в верхней части изображена структурная часть онтологии, в нижней — приведено несколько примеров описания реальных объектов)

Глобальные информационные системы, как правило, состоят из компонентов, территориально значительно удаленных друг от друга, и даже если эти компоненти соеденены через высокоскоростные каналы, скорость их взаимодействия значительно снижается [8]. Потому, выбирая структуру онтологической базы, мы основывались на следующих требованиях: необходимо обеспечить возможность работы не только в рамках общей онтологической базы знаний, но и в режиме локальной системы, у которой фрагмент онтологии, находящийся на удаленном клиенте, представляется как полностью независимая онтологическая структура и может автономно функционировать. Эти два режима могут быть совместны и отличаться только характером запроса к клиенту онтологической базы.

Для поддержки данной функциональной возможности необходимо поддерживать правильную структуру онтологической базы. Как показано на рис. 1, в онтологии представлена информация нескольких типов. Для поддержки нормальной работы системы необходимо дублировать на все клиенты онтологической базы часть онтологии, отвечающую за ее структуру. Для корректной работы системы эти элементы дублируются на все клиены онтологической базы знаний, их модификация возможна только в случае, если изменения были сделаны в синхронизаторе ОБЗ. При этом на синхронизирующем элементе также должны содержаться наборы глобальных объектов, которые отвечают за работу ОБЗ.

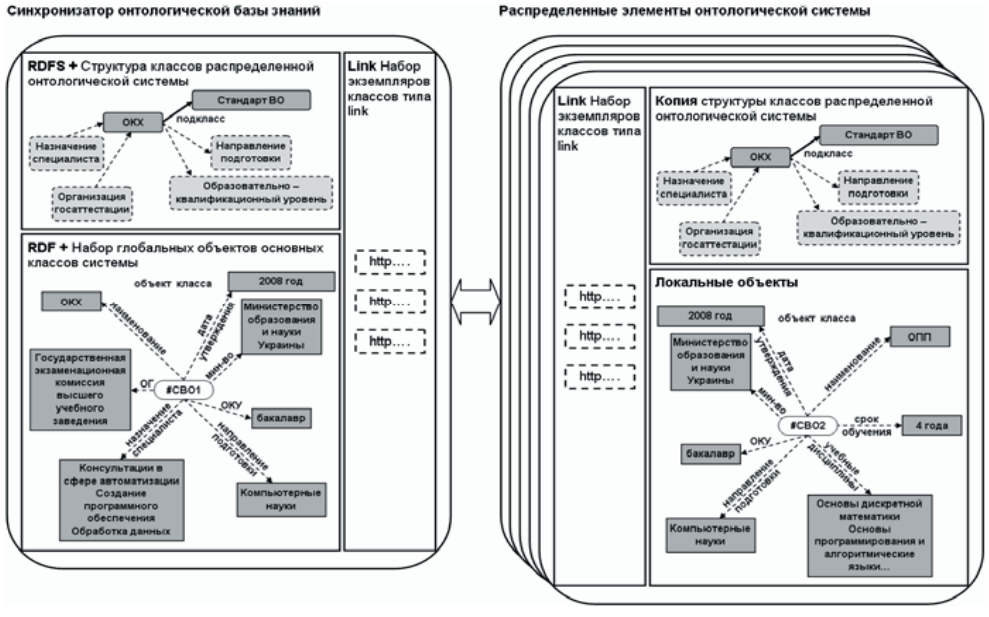

Рис. 2. Принцип построения распределенной онтологии, которая основана на функциональном распределении онтологической базы знаний

На клиентах остается информация о локальных объектах, это позволит легко интегрировать существующие информационные системы организаций с общей онтологической системой, обеспечивая обмен данными между ними. Такое распределение актуально в ситуации, когда элементы онтологической системы располагаются в различных организациях, которые имеют общее подчинение.

2.2. Общий принцип распределения объектов в распределенной онтологической системе

Распределение объектов в онтологической структуре основано на замещении ряда экземпляров класса на один ссылаемый объект, в котором обозначен принцип распределения и находятся ссылки на все доступные компоненты распределенной базы знаний [8]. Все объекты, на которые есть ссылки, и вспомогательные объекты также синхронизируются на все клиенты распределенной онтологической базы. Таким образом, сохраняется целостность работы системы и обеспечивается возможность взаимодействия клиентов с общей информацией. Описанная выше организация онтологической базы позволяет в значительной мере повысить уровень быстродействия и интеграции онтологической системы. Для улучшения процесса интеграции с существующими программными системами и упрощения процесса модернизации и развития онтологической базы знаний был выбран шаблон проектирования Model-View-Controller (MVC).

Шаблон проектирования MVC допускает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может происходить независимо. Модель (Model) предоставляет данные предметной области и реагирует на команды контроллера, изменяя свое состояние. Представление (View) отвечает перед пользователем за отображение предметной области (модели), реагируя на изменения модели. Контроллер (Controller) интерпретирует действия пользователя, уведомляя модель о необходимости изменений. В случае с ОБЗ в качестве модели реализован механизм взаимодействия с онтологической системой, в качестве модуля для взаимодействия исполняющей программной системы с сервером Sezam как контроллера, выступает Web-сервис для взаимодействия с синхронизирующим элементом распределенной онтологической системы и специальный контроллер для локального доступа к онтологической информационной системе. В качестве «представления» выступает локальный Web-клиент, который использует возможность доступа из локальной системы к фрагменту общей онтологической структуры. Благодаря подобной организации появляется возможность легко развивать и модифицировать информационную систему.

2.3. Организация взаимодействия между компонентами распределенной системы через использование Web-сервисов

Важными вопросами организации распределенной онтологической системы являются вопросы безопасности передачи информации и вопросы распределения вычислений в онтологической системе. Для решения этих вопросов на уровне контроллера в модели MVC реализуется Web-севис для взаимодействия с синхронизирующим элементом распределенной онтологической системы. Web-сервис – программная система, которая идентифицируется строкой URI, а общедоступные интерфейсы определены языком XML.

Описание этой программной системы может быть найдено другими программными системами, которые могут взаимодействовать с ней в соответствии с этим описанием с помощью сообщений, основанных на XML, и переданных с помощью Интернет-протоколов. Web -служба является еденицей модульности при использовании сервис-ориентированной архитектуры приложения.

Web-сервис обеспечивает необходимый уровень безопасности переденной информации, а также распределение уровней доступа на уровне клиентов распределенной онтологической системы. Каждая команда Web-сервиса является завершенной транзакцией и в случае обрыва связи гарантирует сохранение целостности базы знаний. Также Web-сервис выполняет функции синхронизации фрагмента онтологии, отвечающего за структуру онтологической базы. В каждый из элементов распределенной онтологической системы (синхронизирующий элемент или клиент онтологической базы) одновременно будут встроены клиентская часть Web-сервиса и сам Web-сервис. Если для вы-

полнения действия необходимо активизировать другой элемент распределенной системы, то используется клиент для отправки вызова и получения ответа от исключенного компонента. Если происходит ожидание вызова от внешних систем, то используется Web-сервис. Организация взаимодействия клиента и сервиса осуществляется с помощью XML-подобного языка, соответствующего протоколу XML-RPC. В случае, если выполняется запрос на передачу фрагментов онтологии, активизируется механизм сериализации объектов с помощью которого осуществляется передача отсутствующих объектов на синхронизирующий элемент.

3. Использование кластеризации сервера баз данных MySQL для распределения даннях

3.1. Организация хранения знаний в онтологических системах

Существует несколько подходов к организации хранения знаний в распределенных онтологических хранилищах. Подход с использованием множества онтологий (рис. 3) использует несколько онтологий, каждая из которых представляет отдельный источник данных. Между онтологиями устанавливаются соответствующие отношения. Запросы к интегрированным данным выполняются на локкальных онтологиях, а координация и распределение запросов происходит с помощью установленных отношений. В такой архитектуре нет необходимости в единой интегрированной онтологии, кроме того, как правило, изменеия в локальных онтологиях не приводят к изменению отношений между онтологиями.

Подход с использованием глобальной онтологии использует интегрированную онтологию, которая описывает данные из всех распределенных источников данных (рис. 4). Все запросы направляются к этой единственной онтологии. Такой подход, хотя и является наиболее очевидным способом организации распределения, требует дополнительных ресурсов: эксперта предметной области, который понимает семантику всех источников данных для определения глобальной онтологии. В качестве источников могут быть использованы отдельные специализированные онтологии [9].

Гибридный онтологический подход (рис. 5) пытается устранить недостатки двух предыдущих подходов. Данные в каждом источнике представлены локальной онтологией, а распределенный словарь, который задается не онтологическим способом, строится для распределения словарей между онтологиями. Главными преимуществами этого подхода по сравнению с предыдущими двумя являются упрощение локального определения онтологий и механизма построения запросов к распределенному словарю.

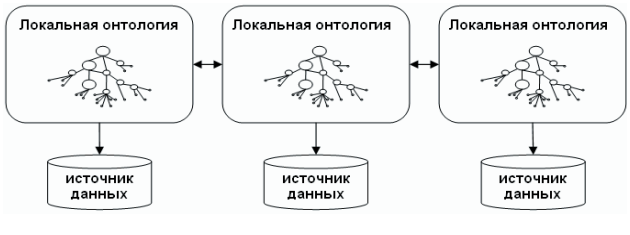

Рис. 3. Архитектура распределения с использованием множества локальных онтологий

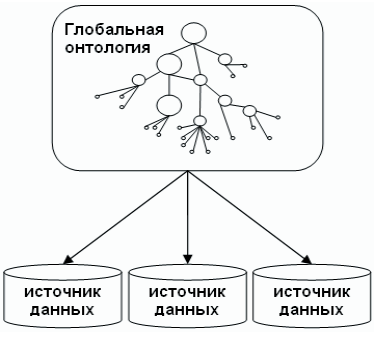

Рис. 4. Архитектура распределения с использованием единой интегрированой онтологии

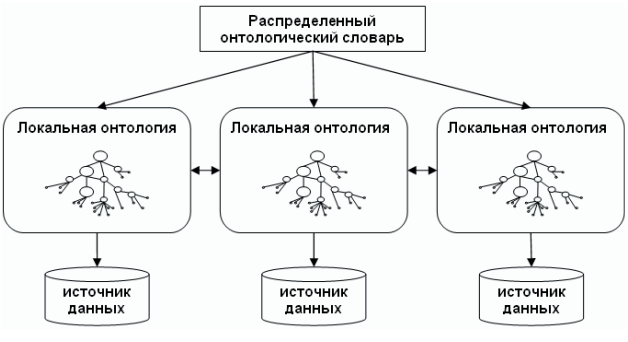

Рис. 5. Архитектура гибридного распределения

3.2. Использование системы Sesame для организации хранения и обработки больших объемов информации

Для разработки распределенных реальных систем (например, онтологического портала менеджмента и оценки ресурсов в области образования и науки) необходимо обеспечить возможность хранения и обработки больших объемов RDF-данных. Одним из решений данной задачи является применение реляционных баз данных совместно с тенологиями Semantic Web. Главным преимуществом такого объединения является то, что оно дает решение проблемы масштабирования с помощью уже разработанных механизмов. На сегодняшний день Sesame может использовать СУБД PostgreSQL, MySQL, Oracle (9i или более современный) и SQL Server. Для накопления данных на уровне хранения и вывода информации в базах данных RDBMS в Sesame используется динамическая схема баз данных, в которой для описания каждого нового класса и его отношений добавляется новая таблица, а отношение «класс-подкласс» определяется по языковой схеме, согласно которой таблица класса является подтаблицей для таблицы, которая описывает суперкласс. Аналогичная схема отношений выполняется для свойств.

Поскольку URI и литералы описываются, как правило, с помощью длинных строк, многие RDF-репозитории, например Sesame, не сохраняют строки в таблицах триплетов, привязывая строки URI к идентификаторам в виде целых чисел таким образом, что данные в общем случае нормализуются в две таблицы: одна таблица триплетов, которая использует идентификаторы для каждой записи, и другая таблица отображений, которая устанавливает соответствия между идентификаторами и строками. К этим двум общим таблицам Sesame добавляет таблицы триплетов и от-

ношений между классами и свойствами. В результате обработки триплетов создаются:

- таблица триплетов (addedTriples), хранит триплеты, которые были добавлены в явном виде за одну транзакцию;
- таблица утверждений, которые были виведены в результате применения одного правила вывода (allInferred);
- таблица триплетов (allNewTriples), хранит все новые триплеты, добавленные за одну транзакцию;
- таблица отношений (direct_subclassof), которая хранит все прямые отношения типа «класс-подкласс»;
- таблица отношений (direct_subpropertyof), которая хранит все прямые отношения типа «свойство-подсвойство»;
- таблица предметной области (domain);
- таблица (instanceof), которая задает отношение «быть экземпляром класса»;
-таблица литералов (literals);
- таблица, которая хранит пространство имен (namespaces);
- таблица триплетов (newTriples), которая хранит новые триплеты, которые были добавлены;
- и другие.

Соединение с MySQL может реализовываться двумя путями:

1) может существовать статический (изначально созданный) репозиторий, который конфигурируется с помощью Configure Sesame для возможности использования MySQL БД;

2) настройка БД может происходить с помощью редактирования исходного кода приложения.

Поскольку Sesame является независимым от СУБД, весь СУБД-специфический код хранится на одном отдельном уровне: уровне хранения и вывода.

3.3. Кластеризация в MySQL

В классической базе данных MySQL данные организованы в виде таблиц, эти таблицы хранятся как файлы на диске сервера баз данных [5]. Кластеризация распределяет обработку данных между несколькими серверами. Кластер серверов позволяет объединить несколько физических серверов (узлов), которые являются партнерами друг друга в процедуре перехода на резервный ресурс. Существует много причин для кластеризации базы данных и несколько различных способов кластеризации.

Вертикальное масштабирование (scaling up) является традационным методом, который применяется в случае, когда используемое оборудование становится устаревшим. Администраторы проводят модернизацию оборудования на уровне используемого сервера или меняют его на более мощный. Горизонтальное масштабирование (scaling out) подразумевает, что добавляется больше серверов, а оборудование остается старым.

Второй способ считается более совершенным с точки зрения надежности и затрат. Однако он является более сложным. Кроме того, и затраты на программное обеспечение являются существенными.

Основные причины кластеризации базы данных:

1) высокий уровень доступности данных. Обеспечение возможности полного отказа отдельных серверов внутри кластера без остановки работы пользователей БД за счет избыточности данных, которые хранятся в кластере;

2) масштабирование. Возможность добавления оборудования к кластеру при соблюдении прозрачности для пользователей БД для повышения эффективности. Это позволяет запускать БД на большом количестве бюджетных компьютеров, и добавлять новые бюджетные компьютеры в случае необходимости.

Кроме того, кластеризация обеспечивает возможность организации отдельных серверов БД с распределенными данными, усовершенствованными наладками репликации и другими кластерами БД, упрощенного менеджмента (например, все узлы в кластере могут контролироваться с одной машины), меньших затрат и большей надежности вцелом. Существует два главных метода кластеризации БД: кластеризация без распределения ресурсов (каждый сервер владеет собственными дисковыми ресурсами), и кластеризация с распределением данных с диска (несколько серверов БД считывают информацию с одного диска).

Для построения кластера необходимо:

– организовать сеть (все узлы кластера должны быть связаны с помощью соединения с минимальной пропускной способностью в 100 Mbps);
– определить количество реплик;
– определить объем RAM-памяти;
– определить объем дискового пространства;
– определить оперативную систему.

MySQL Cluster – это технология, которая делает возможной кластеризацию баз данных, которые находятся в памяти системы, организованной без распределения ресурсов. MySQL Cluster построен таким образом, что каждый компонент системы имеет свою отдельную пам'ять и диск. MySQL Cluster интегрирует стандартный MySQL-сервер с встроенной кластеризированной машиной вывода NDB,, которая работает с хранилищем данных. MySQL кластер состоит из:

– SQL-узлов множества компьютеров, которые запускают MySQL-серверы для получения ответов на запросы. SQL-узлы являются специализированным типом API-узлов, которые описывают приложения обращений к кластеру;

– узлов данных (strorage nodes) для хранения данных, которые содержатся в кластере, и обработки запросов. Количество узлов данных зависит от количества реплик и фрагментов данных;

– одного или нескольких узлов менеджмента (MGM node) или серверов управления, выступающих в роли центральной точки доступа для работы с внутренними кластерами (ролью этого типа узлов является управление другими узлами внутри кластера, выполнение таких функций как представление конфигурационных данных, запуск и остановка узлов, запуск запасного хранения данных и других);

– специализированных программ доступа к данным.

Кластер MySQL интегрирует стандартный MySQL-сервер со встроенной в память машиной вывода NDB, характеризующейся такими качествами как высокая степень доступа и стойкость данных. MySQL-кластер может использоваться с существующими MySQL-приложениями. Такие клиентские приложения посылают SQL-утверждения и получают ответ от MySQL-серверов, которые работают SQL-узлами кластера как отдельные MySQL-сервера. Однако MySQL-клиенты, которые применяют кластер MySQL как источник дан-

нях, могут изменяться для получения преимуществ при работе с разными MySQL-серверами для достижения выравнивания загрузки и перехвата управления в случае отказа. Схема взаимодействия компонентов представлена на рис. 6 [5].

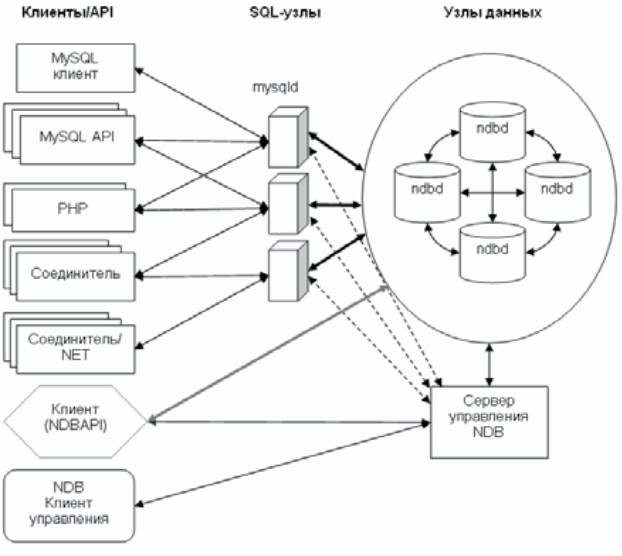

Рис. 6. Схема подключения к кластеру

Репликация – это процесс копирования данных одной БД между другими БД в режиме online для решения задач повышения доступности и надежности хранения данных. Master Server – главный сервер, с которого происходит копирование. Slave Server – подчиненный сервер, на который копируются данные. Все изменения, которые происходят на Master-сервере синхронизируются на Slave-сервере. Slave-серверы с определенной периодичностью опрашивают Master-сервер на предмет изменений в базе. Таким образом, образуется избыточность данных на нескольких серверах.

Важной особенностью является то, что каждый раз переносятся лишь те изменения, которые произошли в данных, а не все данные. Master-сервер записывает все изменения в бинарный журнальный лог, присваивая каждой операции свій номер, и протоколирует ротации бинарных журналов в индексном файле. Когда Slave-серверы обращаются к главному серверу, он сообщает номер последней операции, которая была выполнена, и получает все новые изменения, отсчитывая от этого номера.

MySQL-кластер имеет четыре разных метода получения данных с разными эксплуатационными характеристиками: доступ по первичному ключу, доступ по уникальному ключу, доступ по упорядоченному индексу и просмотр полной таблицы. Доступ по первичному ключу происходит, когда в запросе применяется первичный ключ с использованием поиска в хеше. MySQL-сервер создает хеш первичного ключа и потом, используя алгоритм распределения данных, знает точно, какие узлы данных содержат необходимую информацию и извлекает их, как показано на рис. 7.

Процесс происходит по одному сценарию независимо от количества узлов данных.

Рис. 7 . Доступ по первичному ключу

Доступ по уникальному ключу происходит, когда запрос применяет уникальный ключ для доступа к данным. В этой ситуации происходит поиск по хешу. Однако этот тип доступа является двухшаговым процессом, как показано на рис. 8.

Рис. 8. Доступ по уникальному ключу

Для доступа по упорядоченному индексу MySQL-сервер выполняет операцию, которая называется параллельным сканированием индекса, как показано на рис. 9.

Это означает, что необходимо заставить каждый узел данных просмотреть все индексы, которые хранятся на нем локально. Узел делает это параллельно, а MySQL-сервер комбинирует результаты, когда они возвращаются на сервер.

Последний метод – полного сканирования таблицы – может происходить двумя путями, в зависимости от версии и настроек MySQL-сервера: полное сканирование таблицы без условий и полное сканирование таблицы с условиями.

4. Аппаратно-программная архитектура онтологического портала с распределением онтологии на уровне даннях

4.1. Преимущества распределения онтологического портала на уровне данных

Для распределения онтологического портала менеджмента и оценки национальных образовательных ресурсов Украины песле проведения анализа технологий распределения, которые возможно применить к онтологической системе, была выбрана модель распределения на уровне данных.

Распределение на уровне реализации СУБД решает следующие задачи:

– распределение данных по группам на разных серверах, что снизит нагрузку на носители данных и повысит скорость доступа к хранилищам данных;

– организация балансирования нагрузки системы на нескольких серверах, что повысит ее эффективность работы при подключении нескольких пользователей одновременно;

Рис. 9. Доступ по упорядоченному индексу

– повышение надежности и отказоустойчивости системы за счет дублирования данных.

При реализации такого подхода разработка новых программных модулей в существующей системе не нужна, поскольку кластером СУБД является готовый к использованию существующий продукт. С точки зрения распределенной архитектуры онтологического портала существующая система может стать обычным сервером баз данных, поэтому может быть «прозрачно» включена в существующий прототип портала с выбранной схемой распределения. Также для реализации распределенной версии архитектуры онтологического портала необходимо создать новые требования к администрированию портала в части поддержки кластера СУБД и инфраструктуру серверов для реализации кластера.

4.2. Архитектура кластера системы управления базами данных

Для распределения онтологического портала менеджмента национальных образовательных ресурсов Украины была выбрана организация кластера на основе СУБД MySQL версии 5.1 [5]. Типовая архитектура такого кластера проедставлена на рис. 10.

Інфраструктура кластера содержит в себе следующие типы компонент:

1) MGM Node –узел управления кластером;

2) SQL Nod –узел обработки запросов, обеспечивает виполнение SQL-команд клиентов (в рамках онтологического портала – это сервера поддержки онтологий). Наибольшая загрузка сервера происходит при обработке запросов, это требует повышения требований к скорости и количеству процессоров, а также к объему ОЗУ;

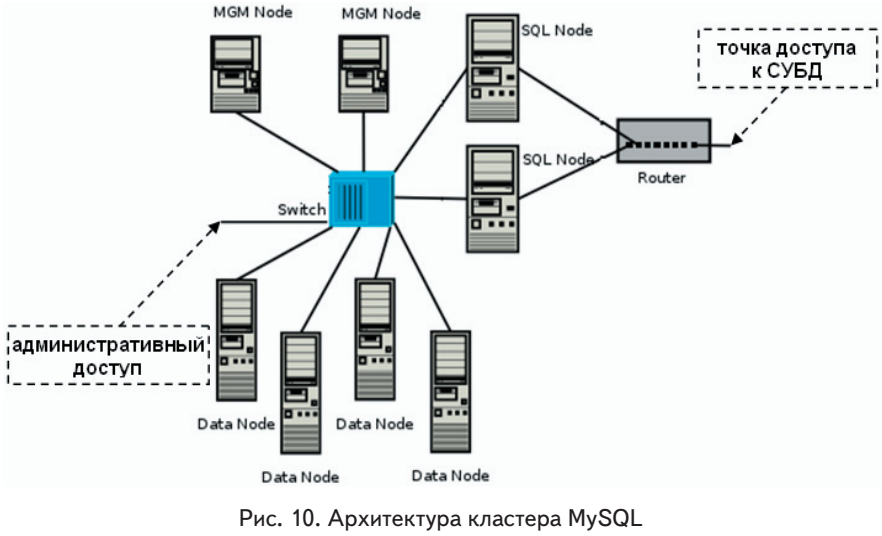

Рис. 10. Архитектура кластера MySQL

3) Data Node – узел данных, который обеспечивает зеркальное (полное копирование данных) или дополнительное (несколько узлов образуют единое пространство данных) хранилище данных. Такой сервер будет загружен при доступе к данным, что повышает требования к системе управления дисками и объемам носителей.

В представленной архитектуре выделяются два типа активного сетевого оборудования:

1) Switch – внутренний коммутатор для связи всех серверов между собой, который должен иметь максимальную пропускную способность и возможность подключения специального клиента для администрирования кластера;

2) Router – маршрутизатор и балансировщик, который выполняет функции ограничения доступа для внешних клиентов к кластеру и распределения нагрузки (путем последовательного перебора маршрутов) на SQL-сервера; должен иметь дополнительные функции маршрутизации.

4.3. Архитектура роспределенного онтологического портала

После проведения анализа существующей системы, схем и моделей распределения онтологических систем и программного обеспечения, которое дает возможность создания и управления распределенными онтологическими системами, была разработана обобщенная архитектура распределенной версии онтологического портала поддержки аккредитации и лицензирования национальных образовательных ресурсов Украины. Разработанная архитектура представлена на рис. 11 [10].

Распределенная архитектура содержит следующие программные сервера: Apache Tomcat – поддержка HTTP-запросов Web-клиентов, сервер статических файлов изображений и документов, сервер приложений для ядра и программных модулей портала; Aduna Sesame – сервер поддержки RDF-графа и онтологической модели данных; MySQL Cluster – набор серверов трех типов, которые реализуются как кластер СУБД.

Структурная схема онтологического портала, которая была разработана на предыдущих итерациях проектирования системы, представлена на рис. 12. В связи с использованием распределения системы при помощи организации кластера СУБД, онтологии модулей системы вынесены в общее хранилище онтологических структур данных Sesame. При этом модификация программного кода будет не нужна, поскольку кластер СУБД имеет такой же SQL-интерфейс, как и автономный сервер СУБД MySQL.

Изменения, которые должны будут произойти в системе при создании ее распределенной модификации, требуют необходимости организации дополнительной конфигурации

репозитория и стека хранения данных в стандартных настройках сервера Sesame.

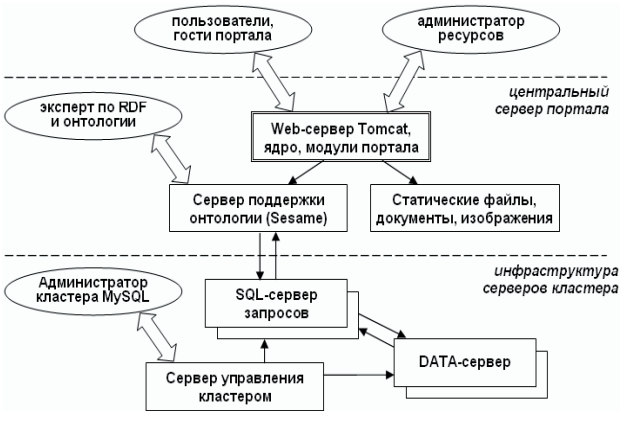

Рис. 11. Архитектура распределенного онтологического портала

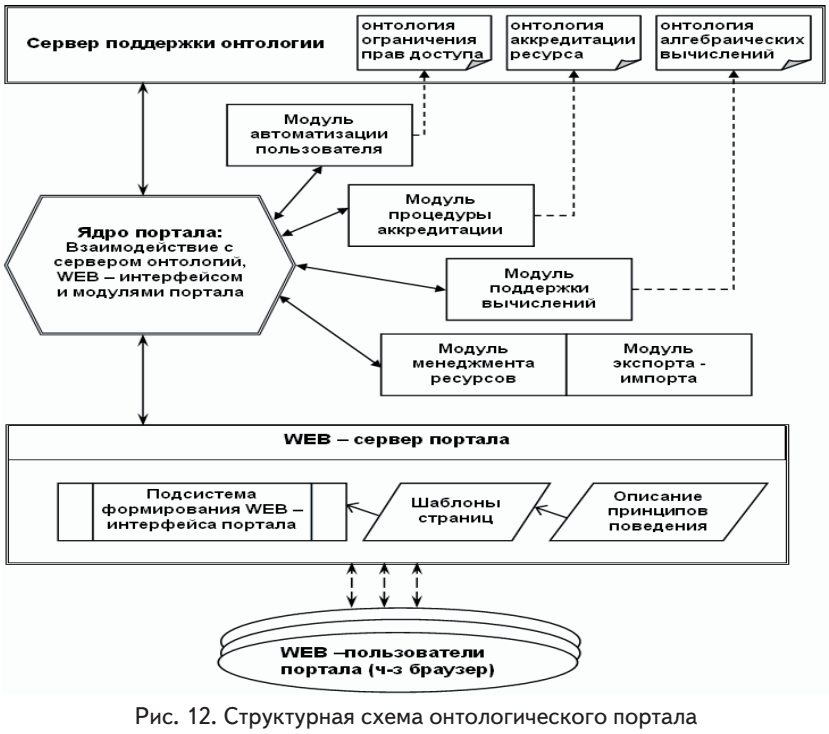

Рис. 12. Структурная схема онтологического портала

Выводы

В результате выполнения работы были получены следующие результаты:

- проведен сравнительный анализ моделей и архитетур распределения информационных систем, которые построены на основе онтологического пожхода; былы рассмотрены модели распределения, которые предусматривают распределение информационной системы как на физическом уровне, так и на уровне бизнес-логики системы;

- проведен анализ современных существующих программных сред, которые удаленно поддерживают обработку онтологических структур; в результате был выбран сервер Sesame; причиной выбора этого сервера хранения и обработки онтологических структур стало то, что существующая версия портала уже построена с его использованием, а также то, что он содержит гибкие механизмы обработки онтологических структур и реализует взаимодействие с сервером MySQL на основе использования Java-машини;

- на основе анализа моделей распределения и программных продуктов, которые необходимо использовать для распределения на онтологическои системы, была выбрана модель распределения на и разработана прораммно-аппаратная архитектура распределенной версии онтологического портала.

Литература

1. OWL Specification Development. [Электронный ресурс] // Режим доступа: www.w3.org/2004/OWL/.
2. FaCT++ Description Logic Reasoner: System Description. [Электронный ресурс] // Режим доступа: www.comlab.ox.ac.uk/people/ian.horrocks/Publications/download/2006/TsHo06a.pdf.
3. DRAGO, System description. [Электронный ресурс] // Режим доступа: http://drago.itc.it/system-description.html.
4. KAON2 – Ontology Management for the Semantic Web. [Электронный ресурс] // Режим доступа: http://kaon2.semanticweb.org/.
5. Официальный сайт разработчика MySQL. [Электронный ресурс] // Режим доступа: 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. [Электронный ресурс] // Режим доступа: http://www.openrdf.org/doc/papers/Sesame-ISWC20-02.pdfs.
7. RDF Specification Development. [Электронный ресурс] // Режим доступа: 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.