ГУАП

КАФЕДРА № 43

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: _____

ПРЕПОДАВАТЕЛЬ:

доцент, к.т.н., доцент	/		/		/	В. Н. Коромысличенко
(должность, учёная степень, звание)		(подпись)		(дата защиты)		(инициалы, фамилия)

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

«Разработать концепт требований к системе хранения онтологической модели структур данных»

ПО КУРСУ: «Разработка и анализ требований»

РАБОТУ ВЫПОЛНИЛ СТУДЕНТ:

4134К	/	Самарин Д.В.
(номер группы)		(инициалы, фамилия)

/		/	
	(подпись студента)		(дата отчета)

Санкт-Петербург 2025

1. Введение

Разработка эффективной системы хранения онтологической модели структур данных становится все более актуальной в условиях динамичного роста информации и необходимости интеграции разнородных источников. Современные информационные системы нуждаются в моделях, способных не только описывать данные, но и обеспечивать семантическую взаимосвязь между объектами, поддерживать логический вывод, автоматическое наследование свойств и быстрый поиск нужной информации. Документы, посвященные онтологиям и графовым базам данных, подчеркивают, что традиционные модели хранения (реляционные, иерархические, сетевые) имеют существенные ограничения, особенно в условиях больших объемов данных и частых изменений. В свою очередь, онтологический подход, реализованный в виде графовой модели, позволяет создать гибкое, масштабируемое и легко модифицируемое хранилище знаний, где новые данные могут добавляться без переработки всей структуры. Кроме того, в одном из документов рассматривается методология многоуровневой трансформации данных, что позволяет интегрировать функциональные, информационные и процессно-сервисные модели для обеспечения комплексного анализа состояния технических объектов.

Цель документа:

Разработать требования к системе, которая хранит, управляет и обрабатывает онтологическую модель структур данных для поддержки принятия решений, аналитики и интеграции информации в условиях динамичного изменения предметной области.

Область применения:

Система предназначена для использования в информационных системах, где важна семантическая структура данных, быстрая обработка запросов, гибкость расширения модели и возможность интеграции разнородных источников (в том числе с использованием подходов виртуальной интеграции данных на основе онтологий).

2. Цели и задачи

Основная цель предлагаемой системы – создание надежного и масштабируемого хранилища онтологической модели, которое способно обеспечить оперативный доступ к знаниям, их интеграцию и дальнейшую трансформацию для поддержки принятия решений. Для этого необходимо решить ряд задач, среди которых ключевыми являются обеспечение гибкости модели, поддержка динамического добавления новых сущностей и отношений, интеграция разнородных источников данных через

виртуализацию и федерирование, а также внедрение механизмов логического вывода и автоматического обогащения информации. Система должна удовлетворять требованиям не только на этапе разработки, но и в процессе эксплуатации, позволяя аналитикам вносить изменения через удобный пользовательский интерфейс без необходимости глубокого вмешательства в программный код.

Основная цель:

Создать гибкую, масштабируемую и отказоустойчивую систему для хранения онтологической модели, которая:

1. Обеспечивает быстрое выполнение семантических запросов (с использованием SPARQL или аналогичного языка).
2. Поддерживает динамическое добавление новых объектов и связей с автоматическим наследованием свойств (например, при использовании RDF-триплетов).
3. Интегрирует возможности анализа и трансформации данных на основе многоуровневых онтологических моделей.

Задачи системы:

1. Реализовать хранение и управление онтологией в виде графовой модели, где сущности и их связи представлены как узлы и ребра.
2. Обеспечить поддержку стандартов RDF, OWL, а также семантических технологий для логического вывода и обогащения данных.
3. Организовать модуль интеграции, позволяющий объединять разнородные источники данных (виртуальная интеграция) с минимальными изменениями в коде.
4. Поддерживать механизм версионности, позволяющий отслеживать изменения в модели.
5. Обеспечить инструменты визуализации и редактирования онтологии (например, с использованием веб-редакторов, подобных Protégé, но интегрированных в систему).

3. Функциональные требования

3.1 Моделирование и хранение онтологии

Система должна представлять онтологию в виде графа, где каждая вершина соответствует классу, объекту или свойству, а ребра отражают отношения между ними. Такой подход позволяет реализовать принцип наследования: если объект обладает дочерним свойством, то он автоматически наследует и свойства родительского уровня. Важно, чтобы хранилище поддерживало стандарты RDF и OWL, позволяющие описывать знания в виде триплетов «объект–отношение–объект», что способствует автоматическому

логическому выводу и уточнению модели. Помимо этого, система должна обеспечивать возможность представления разнотипных и множественных связей, позволяющих описывать сложные иерархические структуры, где один объект может принадлежать нескольким классам и иметь связи с различными типами объектов. Такой подход подтверждается анализом в документе «Онтология и графовые базы данных», где акцент делается на улучшении скорости поиска и удобстве добавления новых данных при использовании графовой модели.

Графовая модель:

- Хранение онтологии в виде графа, где узлы представляют классы, объекты, свойства, а ребра – отношения между ними.
- Поддержка записи данных с помощью RDF-триплетов (объект–отношение–объект) для обеспечения автоматического наследования (например, если объект имеет дочернее свойство, он автоматически получает родительское).

Поддержка стандартов:

- Реализация форматов RDF и OWL для представления знаний и обеспечения семантической совместимости.
- Использование SPARQL или его аналогов для семантических запросов.

Управление сложными связями:

- Поддержка разнотипных и множественных связей между объектами, позволяющих моделировать сложные иерархии и ассоциации.

Логический вывод и трансформация данных:

- Реализация механизма автоматического логического вывода, позволяющего обогащать базу знаний новыми фактами на основе заданных правил и алгоритмов (например, использование SHACL для проверки и трансформации данных).

3.2 Управление данными и версионность

Одним из ключевых требований является поддержка версионности модели. Система должна фиксировать каждое изменение в онтологии, предоставляя возможность отслеживать историю модификаций, сравнивать версии и, при необходимости, откатываться к предыдущим вариантам. Также необходимо обеспечить контроль целостности данных, чтобы каждая операция по добавлению или обновлению информации соответствовала заданным бизнес-правилам. В частности, внедрение механизмов валидации с использованием языков описания ограничений, таких как SHACL, позволит автоматически

проверять корректность и полноту информации. Этот аспект дополнительно проработан в документе, посвященном трансформациям данных о технических объектах, где особое внимание уделяется проверке последовательности трансформаций и соблюдению заданных ограничений.

Версионность модели:

- Поддержка отслеживания изменений в онтологии, управление версиями, возможность отката к предыдущим версиям.

Контроль целостности:

- Механизмы проверки целостности данных и соблюдения бизнес-правил при добавлении или изменении информации.

3.3 Поиск и семантические запросы

Система должна предоставлять мощный механизм семантического поиска, который позволит выполнять сложные запросы по множеству критериев. Запросы реализуются с использованием языка SPARQL или его аналогов, что обеспечивает возможность поиска по связям, атрибутам и логическим правилам, заложенным в онтологическую модель. Важным является оптимизация таких запросов, чтобы время обработки оставалось стабильным даже при больших объемах данных. Анализ литературы показывает, что использование графовых баз данных позволяет достичь фиксированной вычислительной сложности поиска, что особенно критично для интегрированных систем с динамически изменяемой информацией.

Механизм семантического поиска:

- Реализация семантического поиска с использованием SPARQL, позволяющего выполнять сложные запросы по связям и атрибутам объектов.

Оптимизация запросов:

- Использование графовых баз данных для обеспечения фиксированной вычислительной сложности поиска, независимо от объёма данных.

3.4 Интеграция и трансформация данных

В современных информационных системах данные часто поступают из различных источников, имеющих собственные модели и форматы. Система должна обеспечивать виртуальную интеграцию таких данных через модуль, способный выполнять федеративные запросы. Такой подход позволяет объединять информацию, не требуя физического копирования данных, и поддерживать актуальность знаний в режиме реального времени.

Дополнительно, в концепции заложены методы многоуровневой трансформации данных, которые позволяют преобразовывать исходные данные в требуемые представления, разделяя процесс на функциональный, информационный и процессно-сервисный уровни. Документ «Онтологические модели трансформации данных о состоянии технических объектов» подробно описывает подобный подход, что подтверждает необходимость включения таких механизмов в общую архитектуру системы.

Модуль интеграции:

- Виртуальное объединение данных из различных источников посредством механизмов федеративных запросов, адаптеров и посредников (например, с использованием подходов, описанных в Semantic Web технологиях).

Многоуровневая модель трансформации:

- Возможность представления и обработки данных на нескольких уровнях: функциональном, информационном и процессно-сервисном, для обеспечения гибкой трансформации данных.

3.5 Визуализация и редактирование

Для эффективного управления онтологической моделью необходимо наличие удобного инструмента визуализации. Пользователи должны иметь возможность не только просматривать графовую модель, но и редактировать её в интерактивном режиме. Веб-интерфейс системы должен поддерживать построение визуальных представлений, где классы, свойства и отношения отображаются в виде узлов и ребер, с возможностью масштабирования, перемещения и фильтрации элементов. Такой подход позволяет аналитикам и экспертам оперативно вносить изменения в модель, а также отслеживать эволюцию структуры данных.

Графовый редактор:

- Инструменты для визуализации онтологии, позволяющие отображать классы, связи и наследование в виде интерактивного графа.

Веб-интерфейс:

- Пользовательский интерфейс для редактирования, просмотра и фильтрации онтологических моделей, включая возможность динамического добавления новых элементов.

4. Нефункциональные требования

Система должна быть высокопроизводительной и масштабируемой, способной обрабатывать большие объемы данных без существенного замедления при выполнении запросов. Отказоустойчивость является ключевым аспектом: система должна работать в распределенной среде с возможностью кластеризации и репликации, обеспечивая быстрое восстановление в случае сбоев. Важна также безопасность информации: система должна поддерживать механизмы аутентификации, авторизации и шифрования данных, а также вести аудит операций пользователей для обеспечения прозрачности и контроля.

Производительность и масштабируемость:

- Высокая производительность при выполнении семантических запросов даже при больших объёмах данных, с возможностью горизонтального масштабирования.

Отказоустойчивость:

- Возможность работы в распределённой среде с поддержкой кластеризации и репликации, отказоустойчивость и возможность быстрого восстановления данных.

Безопасность:

- Аутентификация, авторизация, шифрование данных, аудит действий пользователей и логирование операций.

Интероперабельность:

- Использование стандартных форматов и API (REST, SPARQL, GraphQL) для интеграции с внешними системами и сервисами.

5. Техническая архитектура

Архитектурное решение предусматривает модульный подход с разделением системы на несколько компонентов. Основное хранилище данных реализуется на базе графовой СУБД (например, Neo4j, OrientDB или TigerGraph), которая хранит онтологию в виде RDF-триплетов. Сервисный слой реализован в виде набора микросервисов, каждый из которых отвечает за определенную функциональность: обработку запросов, выполнение логического вывода, интеграцию с внешними источниками данных и трансформацию информации. Модуль интеграции обеспечивает виртуальное объединение разнородных данных посредством адаптеров, а веб-интерфейс предоставляет пользователям средства для визуализации и редактирования модели. Важно, чтобы все компоненты были развернуты в контейнерах (например, с использованием

Docker) и оркестрированы через Kubernetes, что позволяет добиться гибкости, масштабируемости и быстрой адаптации к изменениям.

5.1 Компоненты системы

Хранилище данных:

- Графовая база данных (например, Neo4j, OrientDB или TigerGraph) для хранения онтологической модели и RDF-триплетов.

Сервисный слой:

- Микросервисная архитектура для обработки запросов, выполнения логического вывода, обработки трансформаций данных и интеграции с внешними источниками.

Модуль интеграции:

- Компонент для виртуальной интеграции данных, обеспечивающий доступ к разнородным источникам с использованием адаптеров и посредников.

Пользовательский интерфейс:

- Веб-редактор для визуализации и редактирования онтологии, а также API для доступа к данным (REST/GraphQL/SPARQL).

5.2 Подход к реализации

Контейнеризация и оркестрация:

- Использование контейнеров (например, Docker) и оркестраторов (Kubernetes) для развертывания компонентов.

Микросервисная архитектура:

- Разделение функциональных модулей (хранение, интеграция, обработка запросов, визуализация) для обеспечения масштабируемости и независимого обновления.

Интеграция с ML-модулями:

- Возможность использования алгоритмов машинного обучения для оптимизации логического вывода, обработки неструктурированных данных и диагностики сбоев.

6. Сценарии использования (Use Cases)

Пользователи системы могут добавлять новые объекты через интуитивно понятный веб-редактор, где вводятся данные о классе, его свойствах и

взаимосвязях с другими объектами. При добавлении нового элемента система автоматически обновляет соответствующие RDF-триплеты и проводит проверку на соответствие заданным ограничениям. В процессе эксплуатации пользователи могут формировать сложные семантические запросы для поиска информации по множеству критериев. Например, можно запросить все объекты, связанные с определенным классом или имеющие заданное свойство, а также получить информацию об их иерархических связях. Администраторы системы имеют возможность управлять версиями модели, отслеживать историю изменений и, при необходимости, возвращаться к предыдущим версиям. Кроме того, аналитики могут подключать внешние источники данных, используя модуль интеграции, что позволяет проводить виртуальное объединение информации и формировать единое представление знаний для поддержки принятия решений.

Добавление нового объекта:

- Пользователь через веб-редактор добавляет новый объект (экземпляр класса) с указанием его атрибутов и связей. Система автоматически обновляет RDF-триплеты, сохраняя наследование свойств.

Выполнение семантического запроса:

- Пользователь формирует сложный запрос (например, «найти все объекты, обладающие свойством A и связанными с объектами класса B через отношение C»), система обрабатывает запрос с помощью SPARQL и возвращает релевантный результат.

Управление версиями онтологии:

- Администратор сохраняет текущую версию онтологии, отслеживает изменения, при необходимости выполняет откат к предыдущей версии.

Интеграция внешних данных:

- Аналитик подключает внешний источник данных через модуль интеграции, выполняется федерирование данных и их виртуальное объединение в онтологию.

7. Риски и ограничения

Как и любая сложная система, данное решение имеет ряд потенциальных рисков. Основным является сложность масштабирования при больших объемах данных – рост количества RDF-триплетов и сложность логических выводов могут потребовать дополнительных оптимизаций. Интеграция разнородных источников данных также сопряжена с проблемами согласования семантики, что может увеличить сроки реализации. Не менее

важным является обеспечение безопасности: защита чувствительной информации требует реализации современных стандартов шифрования, а также регулярного аудита и контроля доступа. Кроме того, сложность системы может привести к повышенной трудоемкости поддержки, особенно в случаях, когда изменения в онтологии затрагивают множество взаимосвязанных компонентов.

Сложность масштабирования:

- Обработка большого объёма RDF-триплетов и сложных графовых запросов может потребовать оптимизации и использования распределённых решений.

Интеграция с внешними системами:

- Требуется тщательное согласование семантики и форматов данных, что может повлиять на сроки реализации.

Обеспечение безопасности:

- Учитывая чувствительность обрабатываемых данных, необходимо реализовать современные стандарты безопасности и регулярно проводить аудит.

8. Заключение

Предложенный концепт требований представляет собой основу для разработки системы хранения онтологической модели структур данных, сочетающей преимущества графовых баз данных с гибкостью онтологического подхода. Внедрение такой системы позволит:

- Обеспечить высокую скорость поиска и обновления данных за счет использования графовой структуры.
- Гибко адаптировать модель к изменениям в предметной области благодаря поддержке динамического добавления новых элементов и связей.
- Интегрировать разнородные источники данных через механизмы виртуальной интеграции, что снижает зависимость от конкретных технологических решений.
- Расширять функциональность системы за счёт модулей логического вывода, трансформации и интеграции с ML-модулями.

Эта объемная концепция может служить отправной точкой для дальнейшей разработки технических спецификаций, архитектурных схем и планов тестирования, адаптированных под конкретные нужды организации. Совмещая результаты анализа обоих документов, концепция отражает как

методологические, так и практические аспекты построения онтологической модели, что позволит создать современную и гибкую систему хранения и обработки знаний.