Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра систем искусственного интеллекта

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Г.М. Цибульский
подпись
«____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – Информационные системы и технологии

Разработка онтологии для оценивания земель сельскохозяйственного назначения

Руководитель _____ доц., канд. техн. наук К.В. Раевич
подпись, дата

Выпускник _____ А.А. Кузнецова
подпись, дата

Красноярск 2019

Продолжение титульного листа бакалаврской работы по теме «Разработка онтологии для оценивания земель сельскохозяйственного назначения»

Нормоконтролер _____ доц., канд. техн. наук К.В. Раевич

подпись, дата

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время в Красноярском крае, в качестве объектов мониторинга выступают лесные, водные объекты, сельскохозяйственные угодья, заповедные и рекреационные зоны и др. В 2017 году в Правительстве Красноярского края была сформирована проектная инициатива «Внедрение технологий дистанционного зондирования Земли в целях повышения эффективности государственного управления».

По одному из соглашений, подписанных на Петербургском международном экономическом форуме - 2019, Красноярский край станет пилотной площадкой по внедрению технологий дистанционного зондирования Земли и цифровизации в государственном управлении. Технологии внедряются госкорпорацией «Роскосмос» в рамках программы «Цифровая Земля» национального проекта «Цифровая экономика РФ», чтобы вести контроль хозяйственной деятельности, оценивать эффективность использования природных ресурсов и т. д. Сервисы должны стать вспомогательным инструментом для администраций регионов.

Онтология для ЗСХН создается и интегрируется на базе лаборатории при кафедре систем искусственного интеллекта Института космических и информационных технологий СФУ, и направлена на создание полноценной базы знаний для системы космического агромониторинга. В процессе разработки на данный момент уже созданы основные модули системы, а также разработан модуль «Онтология агроэкономических показателей ЗСХН». Таким образом целью работы является создание и интегрирование онтологии ЗСХН в систему космического агромониторинга. Для достижения поставленной цели были сформулированы следующие задачи:

1) обзор предметной области;
2) формирование терминологической части и выстраивание иерархии;
3) разработка онтологии предметной области в среде Protégé и интеграция онтологии в СУБД.

Часть 1 Теоретическая глава

1.1 Представление знаний

Представление знаний одна из областей изучения в информатике и в разработке искусственного интеллекта. В информатике эта область заключена в подборе представления конкретных и обобщённых знаний, сведений и фактов для накопления и обработки информации в ЭВМ.

Основной проблемой является то, что не все знания можно легко описать и запрограммировать, т.к. они обычно являются разнородными, многосвязными, часто как декларативными, так и процедурными, связанными не только иерархическими, но и сетевыми структурами. [1]

Одна из проблем в представлении знаний — как хранить и обрабатывать знания в информационных системах формальным способом так, чтобы машины могли использовать их для достижения поставленных задач. Примеры применения — экспертные системы, машинный перевод, компьютеризированное техническое обслуживание и системы извлечения и поиска информации (включая пользовательские интерфейсы баз данных). За историю развития данной области были созданы модели для представления знаний: логические системы, логика предикатов, семантические сети, фреймы, продукции, онтологии. Для систем, основанных на знаниях, «существует» именно то, что может быть представлено. Поскольку сокращение трудоемкости разработки и сопровождения интеллектуальных программных систем является важной проблемой, то использование онтологий для управления знаниями является целесообразным (Шалфеева, 2011).

В исследованиях Computer Vision семантические сети и представления графов использовались для представления пространственных отношений между компонентами изображения. Левин (1978) разработал семантическую сеть, где узлы обозначают объекты, а дуги кодируют пространственные отношения, такие как слева, сверху или сзади. [1]

Семантические сети в данный момент чаще всего используются для лин-

гвистического анализа, в области пространственных знаний семантические сети используются для анализа и модификации знаний в других формах, например для анализа онтологической базы знаний в [2]. Но в первоначальной форме пространственные знания в форме семантических сетей не представляются, так как их функционала недостаточно.

Правила вывода, которые манипулируют знаниями, встроенными в различные представления, представляются в виде продукций: простых модулей, которые ожидают выполнения определенного набора условий и n выполняют некоторое действие. [3] Рассматривая в качестве метода для представления знаний, основанных на здравом смысле, модель TOUR обеспечивает отдельное полезное представление с помощью сбора данных о состоянии частичных знаний. Кроме того, он показывает, как знание ассимилируется из первоначального, очень локального представления в более мощные и глобальные описания. Однако путь заполнения решений нерентабелен в связи с тем, что появляется необходимость большого количества запусков программы.

В искусственном интеллекте было разработано несколько представительных формализмов, обычно основанных на логике (например, Randell и др., 1992), чтобы представлять и обосновывать пространственные отношения.[4] Основная часть формализма предполагает одно примитивное двоичное отношение: C (x; y) читается как `x соединяется с y '. Соотношение C (x; y) является реальным и симметричным. Мы можем дать топологическую модель для интерпретации теории, а именно, что C (x; y) имеет место, когда топологические замыкания областей x имеют общую точку. Введены две аксиомы. – рисунок 1.

$$\forall x \mathrm{C}(x, x)$$
$$\forall xy[\mathrm{C}(x, y) \rightarrow \mathrm{C}(y, x)]$$

Рисунок 1 – Аксиомы для построения дерева интерпретации

Используя данные аксиомы, автор определяет набор отношений для описания предметной области: `DC (x; y) '(` x отключен от y'), `P (x; y) '(` x является часть y '), `PP (x; y) (` x является правильной частью y'), `x = y '(` x совпадает с

y'), `O (x; y) '(` x перекрывает y '), `DR (x; y)' (` x дискретен от y ') `PO (x; y)' (` x частично перекрывает y '), `EC (x; y)' (` x внешне связан с y) ', `TPP (x; y)' (` x является тангенциальной собственной частью y ') и `NTPP (x; y)' (` x не является тангенциальной собственной частью y ') – рисунок 2 и 3.

В данный момент использование предикатов и алгебры отношений обсуждается в основном в определении местоположения, пути, направления. В целом, уже существует богатый набор различных исчислений пространственных отношений, касающихся различных аспектов пространства.

$\mathsf{DC}(x,y)$	$\equiv_{def}$	$\neg\mathsf{C}(x,y)$	//x is disconnected from y
$\mathsf{P}(x,y)$	$\equiv_{def}$	$\forall z(\mathsf{C}(z,x)\rightarrow\mathsf{C}(z,y))$	//x is a part of y
$\mathsf{PP}(x,y)$	$\equiv_{def}$	$\mathsf{P}(x,y)\wedge\neg\mathsf{P}(y,x)$	//x is a proper part of y
$\mathsf{EQ}(x,y)$	$\equiv_{def}$	$\mathsf{P}(x,y)\wedge\mathsf{P}(y,x)$	//x equals y
$\mathsf{O}(x,y)$	$\equiv_{def}$	$\exists\, z(\mathsf{P}(z,x)\wedge\mathsf{P}(z,y))$	//x overlaps y
$\mathsf{PO}(x,y)$	$\equiv_{def}$	$\mathsf{O}(x,y)\wedge\neg\mathsf{P}(x,y)\wedge\neg\mathsf{P}(y,x)$	//x partially overlaps y
$\mathsf{DR}(x,y)$	$\equiv_{def}$	$\neg\mathsf{O}(x,y)$	// x is discrete from y
$\mathsf{EC}(x,y)$	$\equiv_{def}$	$\mathsf{C}(x,y)\wedge\neg\mathsf{O}(x,y)$	// x is externally connected with y
$\mathsf{TPP}(x,y)$	$\equiv_{def}$	$\mathsf{PP}(x,y)\wedge\exists z(\mathsf{EC}(z,x)\wedge\mathsf{EC}(z,y))$	// x is a tangential proper part of y
$\mathsf{NTPP}(x,y)$	$\equiv_{def}$	$\mathsf{PP}(x,y)\wedge\neg\exists z(\mathsf{EC}(z,x)\wedge\mathsf{EC}(z,y))$	// x is a non-tangential proper part of y
$\mathsf{P}^{-1}(x,y)$	$\equiv_{def}$	$\mathsf{P}(y,x)$	// y is a part of x
$\mathsf{PP}^{-1}(x,y)$	$\equiv_{def}$	$\mathsf{PP}(y,x)$	// y is a proper part of x
$\mathsf{TPP}^{-1}(x,y)$	$\equiv_{def}$	$\mathsf{TPP}(y,x)$	// y is a tangential proper part of x
$\mathsf{NTPP}^{-1}(x,y)$	$\equiv_{def}$	$\mathsf{NTPP}(y,x)$	// y is a non-tangential proper part of x

Рисунок 2 – Определение отношений в предметной области

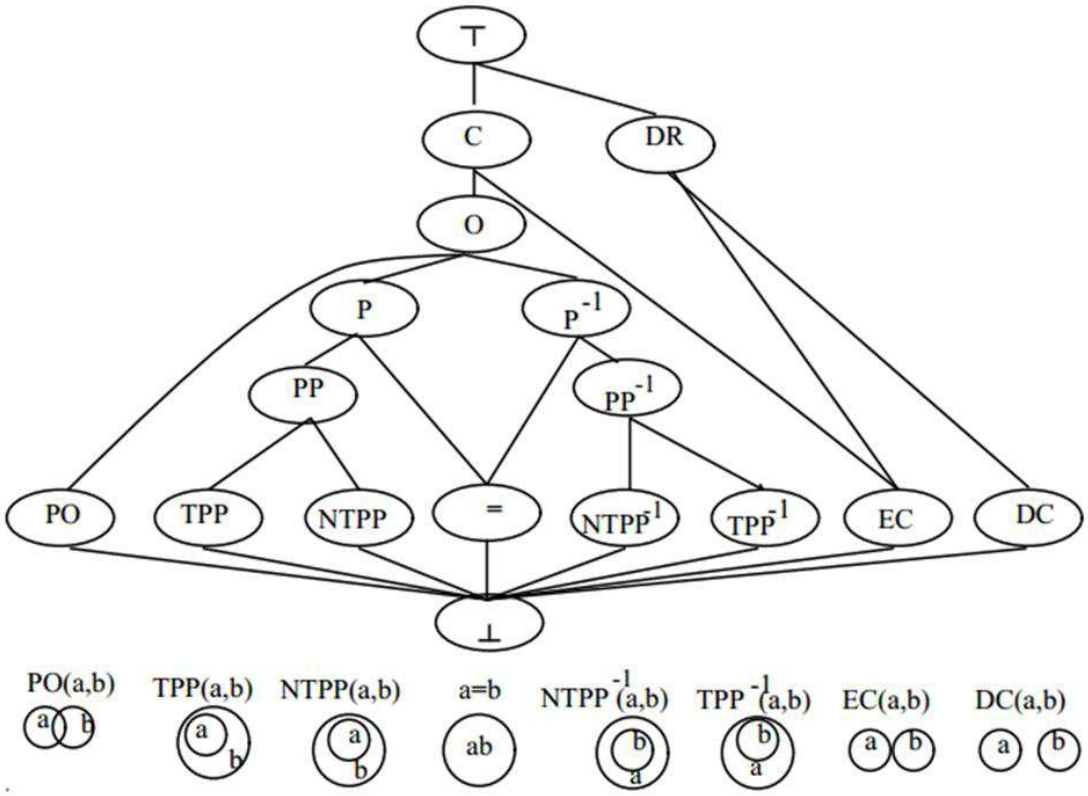


Рисунок 3 – Определение отношений в виде графа

В [5] в общих чертах описаны ключевые исчисления в соответствии с различным типом пространственных отношений. Одни и те же наблюдаемые объекты могут быть абстрагированы как точки или области возрастающей сложности формы при разных разрешениях; поэтому исчисления, моделирующие пространственные отношения между ними, нуждаются в адаптации, чтобы отразить более точное разрешение; чтобы иметь возможность рассуждать на этих разных уровнях, нужно единое исчисление или способ распространения знаний по уровням.

Таким образом, при условии, что системами предикатов сложно описать только один аспект пространственных знаний, то для обширной базы знаний данный способ представления не подходит.

Статья [6] посвящена качественному представлению пространственных знаний и, в частности, представлению бинарных направлений и топологических отношений в двумерном пространстве. Различные системы, основанные на отношениях, связанные с представлением пространственных отношений, были

разработаны в нескольких областях с различными задачами обработки. Графические представления и основанные на логике формализмы были использованы в качественных пространственных рассуждениях. Матрицы пересечений описывают бинарные топологические отношения для множества объектов и вычисляют композиционные отношения и их. Однако и в [5], и в [6] описываются предикаты для онтологических переменных, а также авторами указывается возможность встраивания данных систем в онтологии пространственных данных. Таким образом, более приемлемыми для хранения пространственных знаний являются онтологии.

Использование онтологий способствует созданию адекватных концептуальных моделей, обеспечивая качественное, контролируемое информационное интегрирование. Одна из центральных современных задач геоинформатики – найти способ связи формального представления семантики, заложенной в онтологии, с концептуальными схемами, описывающими информацию, сохраненную в базах геоданных. Главный ожидаемый результат – формальная структура, которая реализует отображение пространственной онтологии в пространственную концептуальную схему.[7]

То есть, онтология, как правило, относится к лексике или системе классификации, которая описывает концепции, работающие в данной области, с помощью определений, которые достаточно подробно описывают семантику этой области. Инженеры-программисты ГИС, которые полагаются на соответствующие онтологии, могут создавать системы, адаптированные к потребностям пользователей. Поскольку онтологически или косвенно онтологические теории и обязательства лежат в основе всех форм познания, онтология — это предприятие, которое пересекает все отрасли науки и информационных систем. Однако на практике это гораздо ближе к вопросам исследований, которые затрагивает географическая информатика, под такими заголовками, как моделирование и представление данных. Однако проект создания онтологий не фокусируется на разработке конкретных алгоритмов и структур данных, которые позволили бы реализовать и кодировать геопространственную информа-

цию и процессы. Скорее он стремится найти соответствующие представления для геопространственных явлений, чтобы соответствовать основным явлениям. [8]

Таким образом, более приемлемым способом организации базы знаний является онтология.

1.2 Понятие онтологии

Онтология является средством формализации знаний о предметной области в форме, упрощающей взаимодействие между пользователем и системой. Термин «онтология» начал использоваться применительно к информационным технологиям в начале 90х годов прошлого века.

Дополнительные преимущества онтологии в вычислительном (сокращается время вычислений) и экономическом (сокращаются затраты на разработку) плане по сравнению с уже существующими решениями, основанными на классических подходах. На самом начальном этапе создания системы всегда возникают вопросы: как именно она будет строиться, какой методологией руководствоваться, какие подходы использовать. Классический подход, с одной стороны, позволяет использовать стандартные методологии, архитектуры и наборы готовых компонентов. Вся информация жестко структурирована и хранится в базах данных. Результат работы такой системы предсказуем и стабилен. С другой стороны, использование закодированных данных и логики делает такие системы сложно расширяемыми и негибкими. В свою очередь, интеллектуальные системы, использующие онтологии в качестве баз знаний, легко поддаются модификации при наличии такой необходимости. Онтологии также позволяют сильно расширить круг задач, решаемых интеллектуальной системой, в различных актуальных направлениях. [9]

Существует достаточно большое количество различных форм и методов описания знаний в виде онтологий со своими достоинствами и недостатками. Также были выделены различные базисы, на которых специалисты могли бы основываться, описывая свои предметные области, так как одной из основных

особенностей онтологического подхода является единство терминологии. Различные формы представления знаний используются в системах, решающих разнообразные задачи, но ни одна из существующих форм не способна полностью покрыть потребности, возникающие при разработке интеллектуальных систем. [10]

Онтология определяет общий словарь для исследователей, которым необходимо обмениваться информацией. Он включает машинно-интерпретируемые определения базовых понятий в предметной области и между ними.

Некоторые из причин для разработки онтологии:

- Распространение понимание структуры информации для людей или программ
- Включение повторного использования знаний предметной области
- Отделение знаний предметной области от оперативных знаний
- Анализ знаний предметной области

Понимание структуры информации людьми или программным обеспечением - одна из наиболее общих целей в разработке онтологий (Musen 1992; Gruber 1993).

Чаще всего формальной моделью онтологии O признают упорядоченную тройку вида O = <C, R, F>, где C – конечное множество концептов т.е. понятий предметной области, определямой онтологией O; R – конечное множество отношений между концептами предметной области ; F – конечное множество функций интерпретации (аксиоматизация), заданных на концептах и/или отношениях онтологии O. Естественными ограничениями, накладываемыми на множество C является конечность и не пустота. Что касается множеств R и F, они могут быть пустыми, что соответствует частным видам онтологии, когда она вырождается в простой словарь и таксономию понятий. [11]

Так как часть модуля «Онтология» уже разработана, углубляться в исследование методов построения онтологий нецелесообразно, в виду того, что дальнейшая работа будет проводиться согласно предыдущим наработкам. Также необходимо рассмотреть, как онтологии встраиваются в существующие

ГИС-системы, чтобы на основании данных примеров спроектировать алгоритм интеграции.

1.3 Обзор методов построения систем основанных на онтологиях

1.3.1 Расширенная визуальная интерпретация

В статье «Использование онтологий интегрированной ГИС» [12] рассматривается подход к онтологиям, использующий основные принципы объектно-ориентированного программирования: наследование, абстракция.

Для анализа изображений необходима система классов/объектов, которая в данном случае выгружается из онтологии, на которой построена система уже в иерархически структурированном виде. Таким образом, данный подход основан на следующих концепциях:

- Для построения ГИС используется комплекс онтологий, в котором общие онтологии (верхнего уровня) не имеющие конкретных экземпляров, и онтологии (нижнего уровня), содержащие всю информацию двух и более онтологий верхнего уровня, уже более конкретные, имеющие экземпляры.
- Каждая онтология верхнего уровня разрабатывается экспертом (группой экспертов), онтологии нижнего уровня собираются пользователями в зависимости от запроса.
- Классификация и кластеризация объектов на изображениях происходит не контролируемо/неконтролируемо, а посредством сегментации изображений и сравнения с объектами онтологии, с присвоением класса (с верхнего уровня до нижнего), в результате объект на изображении по мере классификации приобретет необходимые свойства.
- Объект может относиться к двум и более онтологиям, посредством использования ролей (атрибутов). Например, участок земли имеет две разные специализации (определения, как объекта онтологии), одна для налоговой оценки, а другая для разрешения на строительство.

Результатом процесса классификации является набор изображений, проиндексированных не только по его содержанию, но и по значениям его атрибутов. Изображение рассматривается не только как статический многоугольник со значениями пикселей, но и как набор семантических признаков и соответствующих им значений.

Таким образом, метод расширенной визуальной интерпретации к построению ГИС-систем, основанных на онтологиях, оправдан в случае построения системы с нуля. Однако в данной ситуации он не подходит, так как система уже существует, но способ классификации объектов возможно использовать.

1.3.2 Пространственно-временной спектрально-расширенный метод

В статье «Модель онтологии метаданных наблюдения Земли для пространственно-временного спектрально-семантического обнаружения спутниковых наблюдений» [13] рассматривается подход для эффективного обеспечения детального обнаружения квалифицированных метаданных наблюдения Земли (НЗ) и получения информации о мониторинге влажности почвы на сенсорных изображениях.

По данным Мировой метеорологической службы, для мониторинга влажности почвы доступно до 120 спутниковых датчиков. Если один датчик снимает 1000 изображений в год, всего получается 120 000 изображений. Это слишком большой объем данных для приложения. Кроме того, метаданные наблюдения редко используются в качестве конечного продукта. Трудно получить глубокую информацию из исходных и необработанных данных наблюдений (Kulkarni и McCaslin 2004; Bratasanu, Nedelcu и Datcu 2012).

Онтология метаданных НЗ может использоваться для представления ключевых характеристик наблюдений и для поиска более полезных продуктов НЗ. Предложенная онтология может быть основополагающей в том случае, когда спутниковые датчики предоставляют данные наблюдений (изображения) в течение определенного времени для задачи наблюдения. Характеристики

подхода показаны в следующих аспектах:

● Учитывается процесс наблюдения и ключевые факторы метаданных наблюдения, особенно для пространственно-временных-спектральных характеристик. Онтология обеспечивает систему запросов данных наблюдений, которая может использоваться для поддержки сценариев сенсорной совместимости в приложениях.

● Учитывается тесная связь между приложением и данными наблюдения через пространственно-временную спектральную семантику для фильтрации информации, получаемой потребителями. Это поможет непрофессионалам использовать данные.

Однако метод имеет три ограничения. Во-первых, время, необходимое для запросов увеличивается, когда количество изображений превышает 400. Во-вторых, доступные метаданные запрашиваются без сравнения, когда их время и область перекрываются. В-третьих, критериями поиска с тематической семантикой являются фиксированные параметры.

Таким образом, пространственно-временной спектрально-расширенный метод возможно применить только в случае перманентного, непрерывающегося получения спутниковых данных. В данном случае он неприменим, так как спутниковые данные собираются из открытых источников, поэтому не являются надежными.

1.4 Обзор систем мониторинга

1.4.1 Collect Earth

В статье [14] представлено это бесплатное программное обеспечение с открытым исходным кодом для мониторинга земель, разработанное Продовольственной и сельскохозяйственной организацией Объединенных Наций (FAO). Построенная на Google Desktop и облаке (вычислительная технология), Collect Earth дает доступ к архивам в свободном доступе, содержащим спутниковые изображения, включая архивы с изображениями с очень высоким про-

странственным разрешением (Google Earth, Bing Maps) и с изображениями с очень высоким временным разрешением (например, Google Earth Engine). Опираясь на эти архивы и синергизм изображений нескольких разрешений, обеспечивается инновационный метод для мониторинга земель: дополненная визуальная интерпретация.

Благодаря расширенной визуальной интерпретации, с Collect Earth пользователи могут одновременно анализировать изображения нескольких масштабов и основывать всю свою оценку на одной и той же деятельностьи, которая часто лежит в основе только обучения и части оценки точности исследований LULC. Изменяя входные запросы на сбор информации о Земле, такие как форма сбора данных, дизайн выборки и размер участка, пользователи могут настроить функцию сборки для решения конкретных задач мониторинга земли, таких как восстановление ландшафта, национальные инвентаризации лесов, оценки бедствий и гуманитарной работы, животноводство и управление пастбищными угодьями и т. д. с многовременным и многоуровневым подходом.

Самым значительным нововведением Collect Earth является то, что он позволяет любому вести оценку любой области мира с использованием бесплатных инструментов с открытым исходным кодом, спутниковых снимков VHR свободного доступна онлайн и дополненная визуальная интерпретация.

Таким образом, система Collect Earth основана на методе расширенной(дополненной) визуальной интерпретации, который все также неприменим для данной ситуации, однако метод получения данных путем варьирования запросов вполне подходит.

1.4.2 SemCityMap

Структура, называемая SemCityMap, обеспечивает полный набор функций от обогащения необработанных данных изображений элементарными метками до интеграции методов представления знаний и рассуждений с пользовательскими интерфейсами для высокоуровневых запросов. [15] Особое

преимущество такого подхода заключается в том, что онтологическая информация и аргументация явно интегрированы, так что запросы могут быть сформулированы естественным образом с использованием концепций на соответствующем уровне абстракции, включая дополнительные ограничения.

В предложенной структуре, во-первых, сверточная нейронная сеть (CNN) используется для обеспечения быстрой классификации объектов на карте [16] с использованием заранее определенных элементарных категорий (дороги, здания, деревья и т. д.). Карта с этой первоначальной маркировкой дополняется дополнительной информацией из общедоступных источников для извлечения адресов, ценностей и других знаний высокого уровня. Организовывая эту информацию в онтологиях, структура рассуждений использует комбинацию явно определенных знаний предметной области в форме паттернов онтологии с актуальной ситуационной информацией, полученной из спутниковых изображений.

В работе [17] разработана классификация изображений с помощью онтологии, в которой представлены классификаторы на основе дерева решений и экспертные системы на основе правил. Моделирование с помощью онтологии обеспечивает расширяемую структуру, в которой можно использовать несколько методов классификации изображений.

Благодаря наличию геоданных в общедоступных источниках, таких как OpenStreetMap [18] автоматически извлекались более конкретные метки для классифицированных регионов. Наконец, после генерации экземпляров класса сегмента в онтологии и представления геометрии областей в каждом сегменте вычисляются пространственные отношения между любой парой областей в сегменте.

Таким образом, SemCityMap основана на онтологическом фреймворке для классификации изображений и сверточной нейронной сети. В данном случае такой подход неприменим, так как система уже существует и не является нейронной сетью.

1.4.3 GeoKnow project

Исследования и развитие в области онтологий для ГИС-систем тесно связано с деятельностью по стандартизации, проводимой международными органами, а именно ISO / TC 2112 , OGC3 и W3C4, В Европе директива INSPIRE5 следует открытым стандартам OGC и определяет общие модели данных для ряда доменов приложений, таких как гидрография, защищенные сайты и административные единицы, для повышения функциональной совместимости наборов пространственных данных разных европейских стран. Это обеспечивает правовую и техническую основу для обеспечения совместимости и применимости в трансграничном контексте.[19]

Онтологии использовались в форме таксономий на тематических веб-порталах, но они предоставляют базовые знания, и только в некоторых экспериментальных прототипах они используются для построения поисковых запросов или для группировки результатов поиска в значимые категории. Кроме того, в экспериментальных условиях есть примеры использования OWL для преодоления различий в концептуальных схемах, например [20]. Роль онтологий и инженерии знаний в этих прототипах заключается в основном в предоставлении методологий для интеграции и запросов. Тем не менее, семантическая технология еще не повлияла на управление пространственными данными, а встроенные инструменты ГИС еще не расширены с помощью функции семантической интеграции.

В целом, GeoKnow предоставляет средства для создания связанной семантически аннотированной базы геопространственных знаний, на основе которой можно создавать приложения электронной коммерции, однако данные средства невозможно использовать в данном случае.

1.4.4 Интеллектуальные ГИС на многоуровневых онтологиях

Под интеллектуальной ГИС понимают программно реализованную систему ИИ, которая может наряду с другими использовать в том числе и про-

странственные данные, динамически генерируя соответствующие человеко-машинные интерфейсы. Интеллектуальные ГИС обычно осуществляют динамическую настройку на предметную область, используя системы логического вывода и интерпретацию сценариев обработки, с применением баз онтологий, настраиваемых объектных моделей метаданных, продукционных моделей, методов извлечения знаний, интеллектуального анализа данных, построения разнообразных хранилищ и др. [21]

В статье [22] описывается построение многоуровневых онтологий для ГИС: прикладные онтологии включают два типа пространственных сущностей: классы идентифицируемых объектов (которые могут быть связаны с типовыми объектами изображения) и классы пространственных непрерывных явлений (которые могут быть связаны с серией изображений, моделируемых как поля). Отношение между онтологией изображения и прикладной онтологией устанавливается посредством семантического интерфейса, который выполняет две основные функции:

1) идентифицировать, какая обработка изображения и алгоритмы распознавания образов (описанные в онтологии метода) необходимы, чтобы извлечь желаемые структуры из изображения или преобразовать физические (т. е., пиксельные) значения для получения желаемой информации;

2) отображать понятия прикладной онтологии (т. е., объекты и поля) на структуры, извлеченные из изображения; например, прикладная онтология может содержать понятие железной дороги; используя семантический интерфейс, можно искать идентичные железные дороги среди линейных структур, которые являются частью структурной онтологии изображения.

Проведение четкого различия между изображением и пользовательскими онтологиями позволяет обеспечить два ключевых момента: поддержку множественных представлений для одного и того же изображения и использование изображений для обнаружения пространственно-временных конфигураций географических явлений.

Таким образом, собственно Интеллектуальная ГИС на многоуровневых

онтологиях неприменима, так как необходима только модель онтологии, однако метод построения Интеллектуальных ГИС на многоуровневых онтологиях применим для данной системы, и будет использоваться в качестве метода построения онтологии в данной работе.

1.5 Язык описания онтологии и среда разработки

Для разработки онтологий существует несколько языков описания[23]:

1. OWL — Web Ontology Language, стандарт W3C, язык для семантических утверждений, разработанный как расширение RDF и RDFS;
2. KIF (англ.)русск. (англ. Knowledge Interchange Format — формат обмена знаниями) — основанный на S-выражениях синтаксис для логики;
3. Common Logic (CL) (англ.) — преемник KIF (стандартизован — ISO/IEC 24707:2007).
4. CycL (англ.) — онтологический язык, использующийся в проекте Cyc. Основан на исчислении предикатов с некоторыми расширениями более высокого порядка.

Из представленных языков более всего подходит язык OWL, так как

1. В языках KIF и Common Logic упор сделан на возможность конвертирования знаний, на логические функции, в ущерб наглядности;
2. В языке OWL результатом описания онтологии в большинстве случаев является документ RDF/XML либо OWL/XML, которые могут быть разобраны на теги, что позволит интегрировать онтологию в базу данных.

Среды разработки онтологий [24]:

1. Protégé - бесплатный редактор онтологий с открытым исходным кодом и база знаний (используется язык описания OWL). Разработчик Stanford Center for Biomedical Informatics Research
2. GrOWL - Java приложение для визуализации и редактирования OWL и DL онтологий. Существуют только альфа релизы и плагин для Protege, однако ничего из этого не работает должным образом. Разработка приостановлена.
3. Ontolingua - это система описания онтологий в форме, совместимой с

несколькими языками представления (в основном используется KIF. Предоставляет формы для определения классов, отношений, функций, объектов и теорий.

Таким образом, наиболее подходящей средой разработки является программа Protégé, использующая язык описания OWL.

1.6 Реляционные базы данных, системы управления базами данных и средства для разработки механизма интеграции

Базу данных можно рассматривать как подобие электронной картотеки, т.е. хранилище или контейнер для некоторого набора файлов данных. Между собственно физической базой данных (т.е. данными, которые реально хранятся на компьютере) и пользователями системы располагается уровень программного обеспечения, который можно называть по-разному: диспетчер базы данных (database manager), сервер базы данных (database server) или, что более привычно, СУБД. Все запросы пользователей на получение доступа к базе данных обрабатываются СУБД. Все имеющиеся средства добавления файлов (или таблиц), выборки и обновления данных в этих файлах или таблицах также предоставляет СУБД. [25]

Из СУБД самыми популярными являются клиент-серверные СУБД, которые располагаются на сервере вместе с БД и осуществляют доступ к БД непосредственно, в монопольном режиме. Из СУБД данного типа самыми популярными являются:

1. Oracle Database;
2. Firebird;
3. Microsoft SQL Server;
4. PostgreSQL;
5. MySQL.

Кроме Oracle и Microsoft SQL Server представленные СУБД распространяются свободно. Из 3 оставшихся СУБД в данный момент чаще всего используется либо MySQL, либо PostgreSQL, критичных различий между этими СУБД

нет. Таким образом, для работы был выбран MySQl.

Механизм интеграции будет осуществляться посредством работы с веб-интерфейсом ARC. ARC — это гибкая система RDF для специалистов по семантическим сетям и PHP. Он бесплатный, с открытым исходным кодом, прост в использовании и работает в большинстве сред веб-серверов. Находится в открытом доступе на GitHub. Для установки требуется наличие веб-сервера PHP (5.4 и выше) и MySQL (5.7 и выше). Веб-сервером был выбран пакет Denwer несмотря на то, что последнее обновление было в 2014 году, данная сборка по своим параметрам подходит для реализации пакета ARC, и свободно распространяется.

Так как для работы ARC необходима установка пакета веб-серверов, будет предусмотрена возможность пакетной интеграции онтологии посредством программы, написанной на языке программирования Python. Так как Protégé имеет функцию сохранения в формате XML, то самый быстрый способ работы с базой данных – парсинг XML.

Расширяемый язык разметки, сокращенно XML, описывает класс объектов данных, называемых XML-документами, и частично описывает поведение компьютерных программ, которые их обрабатывают. XML — это профиль приложения или ограниченная форма SGML, стандартного обобщенного языка разметки [ISO 8879]. XML-документы состоят из единиц, называемых сущностями, которые содержат либо неразобранные или проанализированные данные. Анализируемые данные состоят из символов, некоторые из которых образуют символьные данные, а другие - разметки. Разметка кодирует описание логической структуры документа. XML предоставляет механизм для наложения ограничений на структуру хранилища и логическую структуру.[26] Парсинг (Parsing), синтаксический анализ, производится с помощью математической модели сравнения лексем с формальной грамматикой, может быть описан одним из языков программирования. Например, PHP, Perl, Ruby, Python.

Таким образом, для работы с базой данных и написания XML-парсера будет использоваться язык программирования Python. Данный язык выбран в

связи с тем, что имеет готовые библиотеки для парсинга XML, коннекторы для баз данных (включая MySQL).

Выводы по главе 1

Была выполнена одна из задач выпускной квалификационной работы, а именно: проведен обзор предметной области. В результате выполнения обзора были рассмотрены следующие методы построения ГИС-систем на основе онтологии: метод расширенной визуальной интерпретации и пространственно-временной спектрально-расширенный метод. Также были рассмотрены существующие ГИС-системы, основанные на онтологиях: Collect Earth, SemCityMap, GeoKnow, Интеллектуальная ГИС на многоуровневых онтологиях.

Метод расширенной визуальной интерпретации и пространственно-временной спектрально-расширенный метод к построению ГИС-систем оправданы в случае построения системы с нуля. В данной ситуации ни один не подходит, так как система уже существует, но способ классификации объектов из первого возможно использовать. Второй применяется только в случае перманентного, непрерывающегося получения спутниковых данных. В данном случае он неприменим, так как спутниковые данные собираются из открытых источников, поэтому не являются надежными.

Collect Earth основана на методе расширенной(дополненной) визуальной интерпретации, который все также неприменим для данной ситуации. SemCityMap основана на онтологическом фреймворке для классификации изображений и сверточной нейронной сети. В данном случае такой подход неприменим, так как система уже существует и не является нейронной сетью. GeoKnow не является полноценной ГИС-системой, а только предоставляет средства для создания базы знаний, однако при встраивании данной базы знаний в систему космического агромониторинга в будущем могут возникнуть проблемы с лицензией данного продукта. Интеллектуальная ГИС на многоуровневых онтологиях неприменима, так как необходим только модуль онтология, однако метод построения Интеллектуальных ГИС на многоуровневых

онтологиях применим для данной системы, и будет использоваться в качестве метода построения онтологии в данной работе.

Также было выбрано окружение для разработки онтологии: язык OWL, среда разработки - программа Protégé, СУБД – MySQL, средства для интеграции – пакет ARC (и пакет веб-серверов Denwer для его работы) и скрипт на языке Python.

Часть 2 Практическая глава

2.1 Моделирование системы

Существующая технология мониторинга требует привлечения для решения задач предметных экспертов: эксперта в области предметной области, эксперта в области геопространственных данных, эксперта в области обработки изображений. Существующие автоматизированные решения, ориентированные на конечного пользователя, не обладают должной гибкостью, они нацелены на решение задач в узко сформулированной постановке. Тем самым, актуально построение технологии создания информационных систем, позволяющих пользователю формулировать задачи дистанционного мониторинга наземных объектов и искать их решение с использованием интеллектуальных алгоритмов, основанных на знаниях об особенностях исследуемых объектов.

Однако переносимость является проблемой, потому что части общей онтологии могут использовать разные языки и системы представления. В идеале, общие термины должны быть определены на уровне знаний, независимо от конкретных языков представления. Конечно, определения должны быть объединены в некотором обычном формализме, если они должны использоваться в приложениях, основанных на знаниях. Однако нереально или нежелательно требовать, чтобы эти приложения были реализованы на общем языке или системе представления. Это связано с тем, что разным приложениям требуются разные виды рассуждений и языки специального назначения для их поддержки.

Таким образом, проблема переносимости онтологий заключается в поддержке общих онтологий в нескольких системах представления, именно поэтому будет разработана онтология, база данных, отображающая ее, а также механизм интеграции, работающий в обе стороны.

2.2 Моделирование онтологии

Так как онтология создается для оценивания ЗСХН, то основными клас-

сами выбраны почвенные и пространственные характеристики земель, растения и их характеристики, а также характеристики атмосферных явлений, влияющих на ЗСХН. Большую часть классов онтологии составляют культурные растения. Группу полевых культур составляют около 100 важнейших видов, которые дают продукты питания, сырьё для технической переработки и корма для животных. Они различаются по биологическим особенностям, по отношению к условиям окружающей среды, по качеству и количеству получаемой продукции. Общей научной классификацией растений, произрастающих на планете, занимаются специалисты по ботанике. Предпринимались попытки разработать ее также для растений, выращиваемых человеком (культурных), но ни одна из них пока не получила статуса официальной среди агрономов. Существует множество классификаций, предложенных в разное время, рассматривались российские ученые: по Прянишникову [27], Жуковскому [28], Посыпанову [29] и др.

Существует также международная классификация, принятая Продовольственной и сельскохозяйственной организацией ООН (FAO) [30]. Министерство сельского хозяйства и торговли Красноярского края использует Общероссийский классификатор [31] для обозначения культур, выращиваемых на территории края. В этот же список попадает классификация Государственного реестра селекционных достижений, допущенных к использованию [32].

ОКПД 2 построен на основе гармонизации со Статистической классификацией продукции по видам деятельности в Европейском экономическом сообществе (КПЕС 2008) - Statistical Classification of Products by Activity in the European Economic Community, 2008 version (CPA 2008) путем сохранения без изменения кодов и объемов понятий соответствующих позиций. В результате, были составлены классы в соответствии с данными классификациями.

Также в онтологию должны войти сторонние классы типа «Фенофазы» и «Атмосферные явления», содержащие соответствующие экземпляры для описания условий произрастания, включающие статистику и архив реальных данных.

Фенология растений — раздел ботаники, изучающий периодичность в развитии растений в связи с сезонными явлениями в природе. На основе фенологических наблюдений можно дать прогноз интенсивности цветения и обильности урожая плодов.

Переход растений к определенной фазе сезонного развития (фенофазе) связан с рядом изменений в условиях произрастания, главным образом, с температурой, влажностью воздуха и почвы и длиной дня. Также переход фенофаз связан с изменением индекса NDVI, который является важнейшим показателем, который можно рассчитать с помощью снимков ДЗЗ.

Таким образом, база знаний содержит как минимум 6 групп подклассов: агрокультура, агроэкономические показатели, почвенные показатели, пространственные показатели, социально-экономические показатели, атмосферные явления. Реализация данных классов на примере тестовых экземпляров «пшеница» и «Поле_n_000» представлена в виде графа на рисунках 4 и 5.

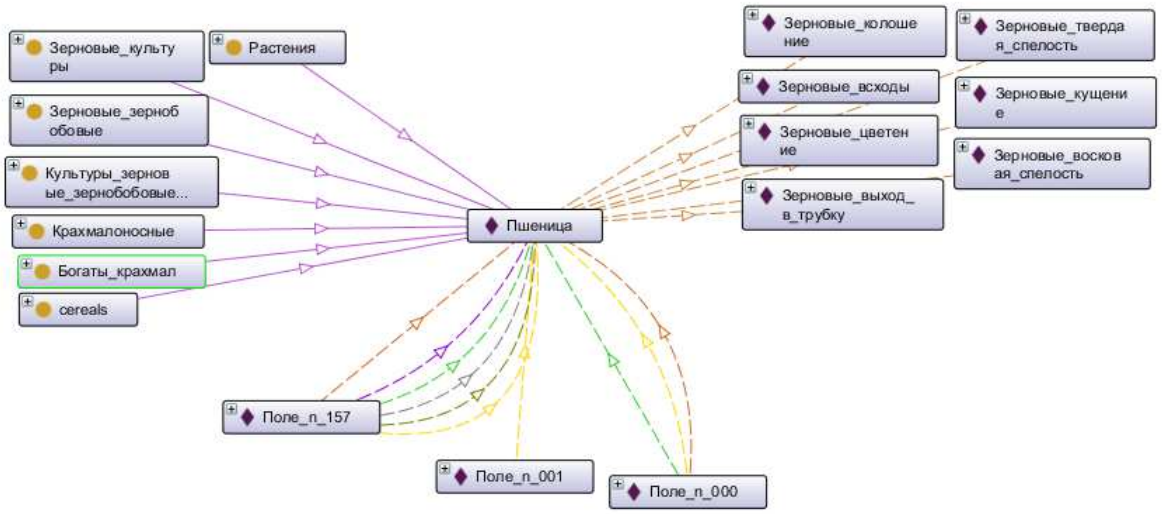


Рисунок 4 – Реализация классов и связей между ними на примере экземпляра «пшеница»

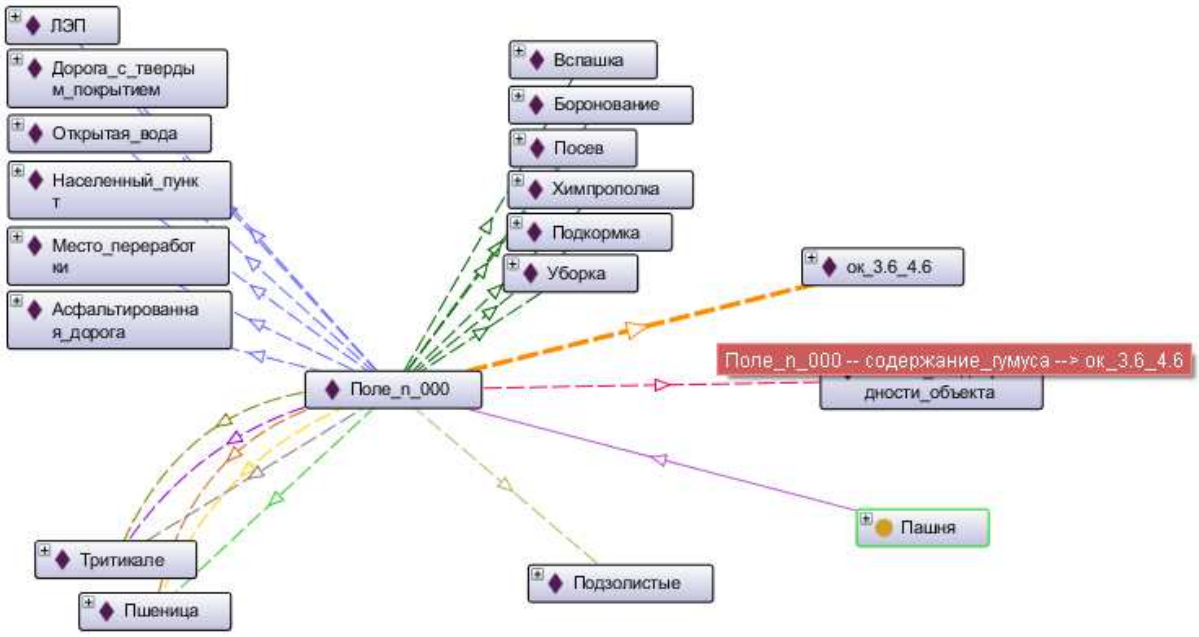


Рисунок 5 – Реализация классов, относящихся к ЗСХН на примере тестового экземпляра «Поле_n_000»

2.3 Наполнение онтологии

Создание классов по намеченной модели проведено успешно, однако в редакторе Protégé добавление свойств объектов (object properties) осуществлено таким образом, что добавить свойство можно только одному объекту, а не всему классу. Поэтому добавление свойств было реализовано через редактирование файлов формата XML (так как Protégé сохраняет онтологию в формате OWL/XML). Аналогично добавлялись свойства данных экземпляра (Data properties). Добавление свойств в этом формате показано на рисунках 6-7.

```
8027    <ObjectPropertyAssertion>
8028        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8029        <NamedIndividual IRI="#Арбуз"/>
8030        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Всходы"/>
8031    </ObjectPropertyAssertion>
8032    <ObjectPropertyAssertion>
8033        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8034        <NamedIndividual IRI="#Арбуз"/>
8035        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Выход_в_трубку"/>
8036    </ObjectPropertyAssertion>
8037    <ObjectPropertyAssertion>
8038        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8039        <NamedIndividual IRI="#Арбуз"/>
8040        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Колошение"/>
8041    </ObjectPropertyAssertion>
8042    <ObjectPropertyAssertion>
8043        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8044        <NamedIndividual IRI="#Арбуз"/>
8045        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Кущение"/>
8046    </ObjectPropertyAssertion>
8047    <ObjectPropertyAssertion>
8048        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8049        <NamedIndividual IRI="#Арбуз"/>
8050        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Твердая_спелость"/>
8051    </ObjectPropertyAssertion>
8052    <NegativeObjectPropertyAssertion>
8053        <ObjectProperty abbreviatedIRI="untitled-ontology-74:находится_в_состоянии"/>
8054        <NamedIndividual IRI="#Арбуз"/>
8055        <NamedIndividual abbreviatedIRI="untitled-ontology-74:Зарастание"/>
8056    </NegativeObjectPropertyAssertion>
8057    <DataPropertyAssertion>
8058        <DataProperty IRI="#биомасса"/>
8059        <NamedIndividual IRI="#Арбуз"/>
8060        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#int"></Literal>
8061    </DataPropertyAssertion>
8062    <DataPropertyAssertion>
8063        <DataProperty abbreviatedIRI="untitled-ontology-74:сорт"/>
8064        <NamedIndividual IRI="#Арбуз"/>
8065        <Literal></Literal>
8066    </DataPropertyAssertion>
```

Рисунок 6 – Свойства объекта в формате XML

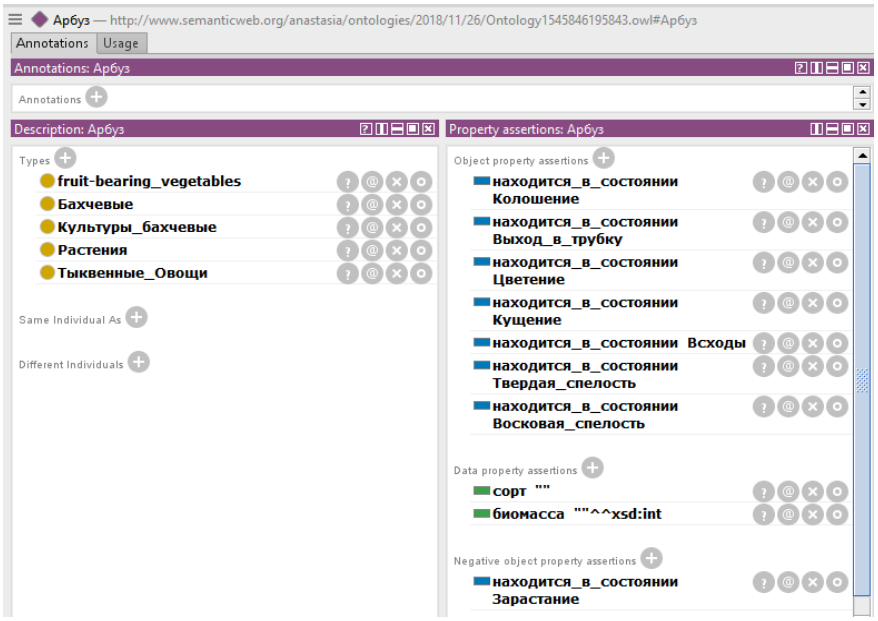


Рисунок 7 – Свойства объекта в Protégé.

Таким образом, в результате наполнения онтологии было создано 286 классов и 505 экземпляров, количество остальных элементов представлено на рисунке 8.

Ontology metrics:

Metrics	
Axiom	2581
Logical axiom count	1695
Declaration axioms count	874
Class count	286
Object property count	22
Data property count	60
Individual count	505
Annotation Property count	2

Рисунок 8 – Метрики онтологии (количество элементов)

2.4 Создание запросов к онтологии

Для онтологии оценивания ЗСХН необходима возможность подбора ЗСХН по параметрам пользователя, то есть необходим механизм, с которым будет работать пользователь и который не требует трудоемких манипуляций.

Функционал Protégé позволяет создавать и исполнять запросы к онтологии, что позволяет решать задачи поиска необходимых знаний по определенным условиям. Существует два вида запросов: SPARQL и DLquery.

2.4.1 SPARQL запросы

RDF - это ориентированный формат помеченных графовых данных для представления информации в Интернете. SPARQL содержит возможности для запроса обязательных и необязательных графовых шаблонов, а также их соединений и дизъюнкций. SPARQL также поддерживает агрегацию, подзапросы, отрицание, создание значений с помощью выражений, тестирование расширяемых значений и ограничение запросов по исходному графику RDF.[33] SPARQL 1.1 - это набор спецификаций, которые предоставляют языки и протоколы для запроса и управления содержимым графа RDF в Интернете или в хранилище RDF. Результаты запросов SELECT в SPARQL включают пакеты отображений из переменных в термины RDF, которые часто удобно представлены в табличной форме. [34] Синтаксис языка SPARQL аналогичен SQL и представлен на рисунке 9. Полное описание данного синтаксиса представлено в [33] вместе с примерами.

```
PREFIX значение: <IRI>
SELECT [DISTINCT] [REDUCED]?переменная(столбец) / *(все значения)
    WHERE {
            ?субъект свойство ?объект.
            [ VALUES ?объект {} ]
            [ OPTIONAL { ?субъект свойство ?объект} ]
            [ FILTER [ NOT EXIST {} ] [ regex(?объект, "маска/значение") ] ?объект условие ]
            [ BIND ( [IF ()] [CONCAT () ] AS ?объект_ ]
            [ UNION {} ]
            [ MINUS {} ]
    }
    [ ORDER BY ]
    [ DESC (?объект) ]
    [ OFFSET количество ]
    [ HAVING () ]
```

Рисунок 9 – синтаксис запроса SPARQL

Таким образом, с помощью SPARQL запроса можно ставить ограничения на диапазон значений Data property(свойства данных), и выбирать те экземпляры, которые обладают нужными Object property(объектными свойствами). Например, для выбора ЗСХН относящегося к классу «Пашня», на котором проводилась агротехническая обработка (подкормка) и имеющему значения показателя «Гумус» меньше 5% будет соответствовать следующий запрос (рисунок 10).

```
SPARQL query:
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ont: <http://www.semanticweb.org/anastasia/ontologies/2019/4/26/untitled-ontology-55#>
SELECT distinct ?individual ?number
WHERE {
            ?individual rdf:type owl:NamedIndividual.
            ?individual rdf:type ont:Пашня.
            ?individual ont:агротехническая_обработка ont:Подкормка.
            ?individual ont:Гумус ?number . FILTER (?number<5.0)
}
```

individual	number
Поле_n_000	"4.6"^^<http://www.w3.org/2001/XMLSchema#float>

Рисунок 10 – Запрос к онтологии на языке SPARQL

2.4.2 DL query запросы

Вкладка DL Query предоставляет мощную и простую в использовании

функцию поиска в классифицированной онтологии. Он поставляется со стандартным дистрибутивом Protégé Desktop (версии 4, 5 и выше). Язык запросов (выражение класса), поддерживаемый плагином, основан на синтаксисе OWL Manchester, удобном для пользователя синтаксисе OWL DL, который в своей основе основан на сборе всей информации о конкретном классе, свойстве или индивидууме в единую конструкцию, называемую фрейм.[35] Выполнить запрос можно только по классифицированной онтологии, для этого в комплекте с Protégé поставляются Reasoner'ы FaCT++ и HermiT, также есть возможность установить дополнительный Reasoner, если возникнет необходимость. Данные запросы можно сохранить в онтологии, и они легкодоступны для пользователя. Например, при необходимости выбрать ЗСХН по параметрам «содержание гумуса» и «вид почвы» пользователь может создать запрос, который представлен на рисунке 11. В Protégé встроена система подсказок, что позволяет малознакомому с онтологией пользователю строить запросы.

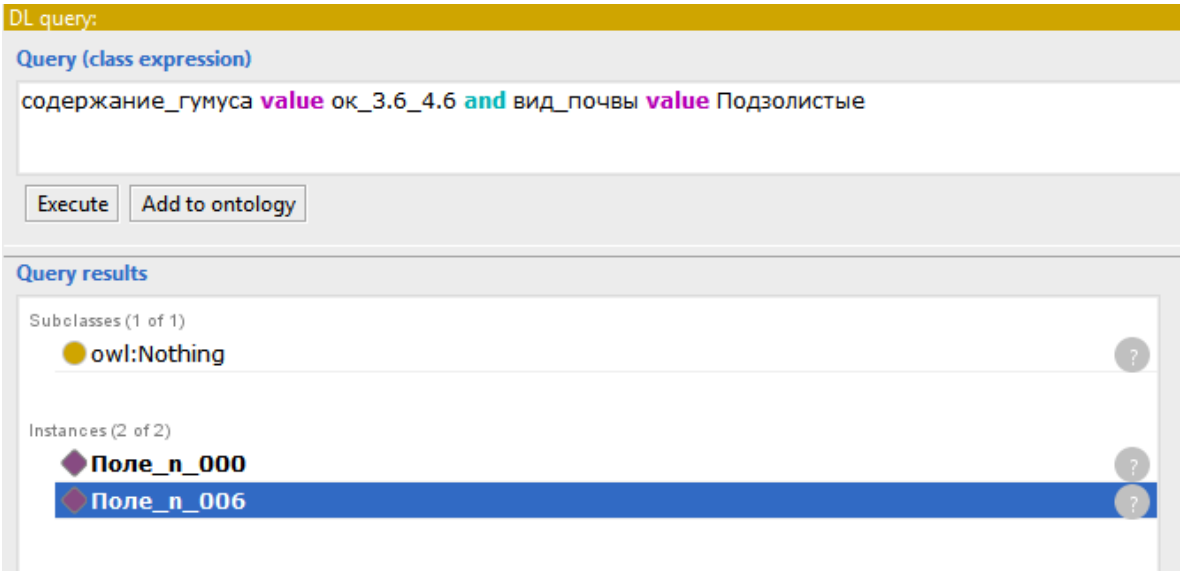


Рисунок 11 – Запрос на поиск ЗСХН по параметрам

2.5 Интеграция онтологии в СУБД

Операция конвертации базы данных в онтологию отражена в работах [36] и [37], [38]. В статьях [36] и [37] приводится пример работы с Jena API и устаревшими версиями Protégé. Поэтому такие подходы для отображения также не

подходят, однако правила отображения, изложенные в данных работах, актуальны.

В статье [38] рассматриваются сторонние конвертеры и запускаемые неуправляемые скрипты, что не позволяет настраивать гибкое отображение онтологии и следить за выполнением процесса. Однако для Protégé разработано несколько плагинов для отображения СУБД, таких как: Ontop, OWL2ToRDB, Mastro OBDA. Из этих плагинов в данный момент более документированным и удобным в употреблении является Ontop (Ontop Mapping).

Таким образом, для интеграции СУБД в онтологию используется база данных на сервере MySQL и плагин для Protégé – Ontop Mapping.

Операция отображения онтологии в формате OWL в формат объектной/реляционной базы данных рассматривается в работах [39], [40]

В статье «Конвертер онтологической базы знаний станков и деталей машиностроительного производства в объектную базу данных на основе ANTLR» [39] приводится пример конвертера в объектную базу данных, но так как в системе изначально используется реляционная база данных, такой подход неприменим в данной ситуации. Также существует скрипт конвертации в базу данных [40], однако он актуален только для Protégé версии 3.x, в более новых версиях, данный конвертер не работает.

Для отображения онтологии был выбран пакет ARC2. Данная система может автоматически создать набор RDF троек из онтологии и мигрировать эти данные в базу данных (используется MySQL, таблицы создаются автоматически). Так как пакет требует пакета веб-серверов, настройки и правки PHP кода, он довольно сложен в установке. В связи с этим было принято решение разработать дополнительно механизм интеграции, работающий в пакетном режиме.

Для того, чтобы программа могла работать с базой данных, необходимо построить ее структуру и явно включить в скрипт. Физическая модель базы данных представлена на рисунке 12.

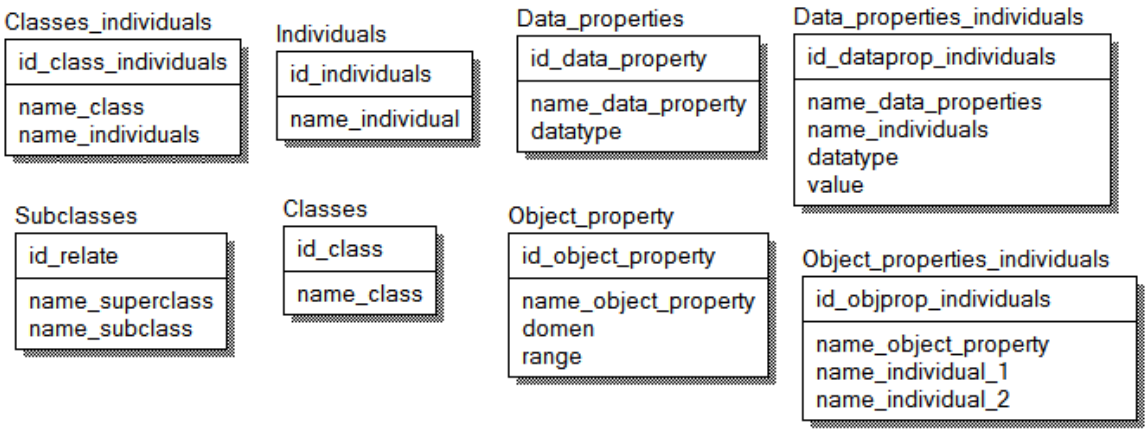


Рисунок 12 – Физическая модель базы данных онтологии для отображения

На рисунке 12 показаны таблицы базы данных (названы по-английски, чтобы избежать проблем с несовпадением кодировок, возникающих довольно часто при использовании кириллицы):

- Classes – отображение классов, основное поле – название;
- Data_properties – отображение свойств данных, содержит поля название свойства и тип данных;
- Object_property – отображение объектных свойств (связей между экземплярами классов), содержит поля название, domain – это набор классов, к экземплярам которых применимо данное свойство, range - тип значений, которые свойство может принимать;
- Individuals – отображение экземпляров классов, основное поле – название;
- Sublasses – отображение связи класс-подкласс, содержит поля название надкласса и название подкласса;
- Classes_individuals – отображение связи класс-экземпляр, содержит поля название класса и название экземпляра;
- Data_properties_individuals – отображение связи экземпляр-свойство данных, содержит поля название свойства данных, название экземпляра класса, тип данных, значение;

- Object_properties_individuals – отображение связи экземпляр- объектное свойство, содержит поля название объектного свойства, название первого экземпляра и второго, которые связывает объектное свойство.

После разработки базы данных был разработан собственно исполняемый модуль – скрипт. Алгоритм его работы основан на уже существующей библиотеке для парсинга XML: документ XML, подающийся на вход обрабатывается парсером, затем анализируется и найденные данные вставляются в базу данных. Подключение к базе данных прописано в коде скрипта, поэтому для того, чтобы изменить свойства подключения, необходимо их явно указать. Пример основной логики представлен на рисунке 13. Полный код скрипта представлен в приложении А.

```
def handleClass(text):
    classes = []
    cl = []
    cl = text.getElementsByTagName('Class') #находим тег по названию
    for elem in cl:
        fixed = elem.attributes['IRI'].value #находим атрибут «IRI» -название
        l1 = fixed.find('#')
        fixed = fixed[l1+1:] #обрезаем название- в Protégé в названиях ссылки
        classes.append(fixed)
        listcl=list(set(classes))
    try: #попытка подключения к бд и вставка найденных данных(классов)
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for el in listcl:
                sql = "INSERT IGNORE INTO classes (name) VALUES(%s);"
                cursor.execute(sql,el)
                connection.commit()
    finally:
        connection.close()
```

Рисунок 13 – основная логика скрипта

Выводы по главе 2

В результате моделирования онтологии выявлено, что база знаний должна содержать как минимум 6 групп подклассов: агрокультура, агроэкономические показатели, почвенные показатели, пространственные показатели, социально-экономические показатели, атмосферные явления. Данная модель реализована в Protégé, и в результате наполнения онтологии было создано 286 клас-

сов и 505 экземпляров, количество остальных элементов представлено на рисунке 8.

Для доступа к знаниям, содержащимся в онтологии созданы запросы в среде SPARQL, в которой можно ставить ограничения на диапазон значений Data property (свойства данных), и выбирать те экземпляры, которые обладают нужными Object property (объектными свойствами).

А также были созданы запросы DL query, выполнить которые можно только по классифицированной онтологии, для этого в комплекте с Protégé поставляются Reasoner'ы FaCT++ и HermiT. Так как в Protégé встроена система подсказок, что позволяет малознакомому с онтологией пользователю строить запросы, данные запросы будут строиться именно пользователями и в онтологию не включены.

Также был опробован механизм для интеграции СУБД в онтологию (используется база данных на сервере MySQL и плагин для Protégé – Ontop Mapping). И так как выбранное диалоговое решение (ARC2) требует от пользователя настройки путем редактирования кода (php) и пакета веб-серверов, разработан дополнительный механизм интеграции, работающий в пакетном режиме, на языке программирования Python.

ЗАКЛЮЧЕНИЕ

Результатом выполнения выпускной квалификационной работы является разработанная онтология для оценивания земель сельскохозяйственного назначения, а также механизм ее интеграции с СУБД.

Для достижения поставленной цели был проведен обзор предметной области, в ходе которого был определен метод построения онтологии, Интеллектуальная ГИС на многоуровневых онтологиях не строилась, однако метод построения онтологии использовался в данной работе.

В результате моделирования онтологии выявлено, что база знаний должна содержать 6 групп подклассов: агрокультура, агроэкономические показатели, почвенные показатели, пространственные показатели, социально-экономические показатели, атмосферные явления. Данная модель реализована в Protégé, и в результате наполнения онтологии было создано 286 классов и 505 экземпляров.

Для доступа к знаниям, содержащимся в онтологии созданы запросы в среде SPARQL. Также был опробован механизм для интеграции СУБД в онтологию (используется база данных на сервере MySQL и плагин для Protégé – Ontop Mapping). И так как выбранное диалоговое решение требует от пользователя настройки путем правки кода и пакета веб-серверов, разработан дополнительный механизм интеграции, работающий в пакетном режиме, на языке программирования Python.

Разработанная онтология будет в дальнейшем применяться в НУЛ «ИПКМ» Института космических и информационных технологий на кафедре систем искусственного интеллекта СФУ.

Работа представлена на Международной научно-технической конференции студентов, аспирантов и молодых ученых «Проспект Свободный -2019». А также статья по результатам бакалаврской работы будет представлена на VI Международной научной конференции «Региональные проблемы дистанционного зондирования Земли» (РПДЗЗ – 2019).

Онтология оценивания ЗСХН разработана при выполнении бакалаврской работы по направлению 09. 03. 02 «Информационные системы и технологии», по профилю подготовки 09. 03. 02. 05 «Информационные системы в административном управлении».

СПИСОК СОКРАЩЕНИЙ

1) СУБД – система управления базами данных, стр. 6;
2) ЗСХН – земля сельскохозяйственного назначения, стр. 6;
3) ГИС-система – геоинформационная система, стр.16;
4) НЗ –Наблюдение Земли, стр.16;
5) LULC – land use/land cover – землепользование, земной покров, стр.17;
6) VHR – Very High Resolution – снимки высокого разрешения, стр. 18;
7) OGC – Open Geospatial Consortium – организация по стандартизации в сфере геопространственных данных и сервисов, стр. 19;
8) W3C – World Wide Web Consortium – консорциум всемирной паутины, стр. 19;
9) ИИ – искусственный интеллект, стр.20;
10) FAO – Food and Agricultural organization – Продовольственная организация ООН, стр. 28;
11) ОКПД – Общероссийский классификатор продукции по видам деятельности, стр. 28;
12) NDVI – Normalized Difference Vegetation Index — нормализованный относительный индекс растительности, стр. 29;
13) SPARQL – рекурсивный акроним от англ. SPARQL Protocol and RDF Query Language — язык запросов к данным, представленным по модели RDF – стр.32

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A knowledge-based computer vision system. // Levine, M. In: Hanson, A. and Riseman, E., eds. Computer Vision Systems. New York: Academic Press, 1978, pp. 335-352.

2. Меры важности концептов в семантической сети онтологической базы знаний // Карпенко А.П., электронное научное издание Наука и образование, №7, 2010 [электронный ресурс] – режим доступа: http://technomag.edu.ru/doc/151142.html

3. Modelling spatial knowledge // Benjamin Kuipers, Cognitive Science Volume 2, Issue 2, April–June 1978, Pages 129-153

4. Randell, D.A., Cui, Z., and Cohn., A. A spatial logic based on regions and connec-tion. Proceedings of the Third International Confi;rence on Principles of Knowledge Representation and Reasoning; 1992.

5. A Survey of Qualitative Spatial Representations // Chen, J, Cohn, AG, Liu, D et al. , Knowledge Engineering Review, 30 (1), 2013 – с. 106 - 136.

6. Qualitative Representation of Spatial Knowledge in Two-Dimensional Space // Dimitris Papadias and Timos Sellis, VLDB Journal,3, 1994, с. 479-516

7. Обработка пространственной информации // И.Н.Розенберг, Перспективы Науки и Образования. 2015. 3 (15) - С. 17-24

8. Ontological Foundations for Geographic Information Science // David Mark, Max Egenhofer, Stephen Hirtle, Barry Smith, A Research Agenda for Geographic Information Science, 2004 - pp.335-350

9. Методы и средства построения онтологий для визуализации связанных информационных объектов произвольной природы в сложных информационно-аналитических системах // М. М. Матюшин, Т. Г. Вакурина, В. В. Котеля, «Информационно-управляющие системы» №2, 2014, 17 с.

10. A Translation Approach to Portable Ontology Specifications // Thomas R. Gruber, Knowledge Acquisition, 5(2):199-220 с., 1993

11. Онтологии деятельности для ситуационного управления предприятиями в реальном времени // П.О. Скобелев - “Онтология проектирования”

научный журнал, 1(3)/2012 , 2012– 6-38с.

12. Using Ontologies for Integrated Geographic Information Systems // Frederico T. Fonseca, Max J. Egenhofer, Peggy Agouris - Transactions in GIS6(3), 2002 – с. 231 – 257

13. Earth observation metadata ontology model for spatiotemporal-spectral semantic-enhanced satellite observation discovery: a case study of soil moisture monitoring // Xiaolei Wang, Nengcheng Chen, Zeqiang Chen, Xunliang Yang, Jizhen Li – Journal GIScience & Remote Sensing Volume 53, 2016 - Issue 1, с. 22-44

14. Collect Earth: Land Use and Land Cover Assessment through Augmented Visual Interpretation // Adia Bey and co-authors, Remote Sens. 2016, 8, 807 – с. 24

15. An Ontology-Based Reasoning Framework for Querying Satellite Images for Disaster Monitoring // Marjan Alirezaie, Andrey Kiselev, Martin Längkvist, Franziska Klügl, Amy Loutfi

16. Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks // Längkvist, M.; Kiselev, A.; Alirezaie, M.; Loutfi, A., Remote Sens. 2016, 8, с. 329.

17. A Framework for Ocean Satellite Image Classification Based on Ontologies // Almendros-Jimenez, J.M.; Domene, L.; Piedra-Fernandez, J.A. - IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2013, 6, с. 1048–1063

18. OpenStreetMap. 2017. [Электронный ресурс]: режим доступа - https://www.openstreetmap.org/export#map=5/51.500/-0.100

19. Managing Geospatial Linked Data in the GeoKnow Project // Jens Lehmann, Spiros Athanasiou , Andreas Both , Alejandra Garcia Rojas , Giorgos Giannopoulos , Daniel Hladky , Jon Jay Le Grange , Axel-Cyrille Ngonga Ngomo , Mohamed Ahmed Sherif , Claus Stadler , Matthias Wauer , Patrick Westphal , Vadim Zaslawski, Volume 20: The Semantic Web in Earth and Space Science. Current Status and Future Directions, p. 51-78

20. Catherine Dolbear and Glen Hart. Ontological bridge building – using ontologies to merge spatial datasets. In AAAI Spring Symposium: Semantic Scientific Knowledge Integration, pages 15–20. AAAI, 2008.

21. ГИС сегодня: состояние, перспективы, решения // Ю. И. Шокин1, В. П. Потапов - Вычислительные технологии, Том 20, № 5, 2015, с. 175-213

22. О многоуровневой онтологии геоданных // С.К. Дулин, С.В. Духин, В.Г. Поповидченко, Системы и средства информ., 2007, выпуск 17, 337–354

23. Языки описания онтологий для технических предметных областей // Е. Е. Буракова, Н. М. Боргест, М. Д. Коровин - Вестник Самарского государственного аэрокосмического университета, № 3(45) 2014, с. 144-158

24. Обзор инструментов инженерии онтологий // Овдей О.М., Проскудина Г.Ю., Киев – 2005г., 10 с.

25. Введение в системы баз данных // К. Дж. Дейт - 8-е изд. — М.: Вильямс, 2006. — С. 1328.

26. Расширяемый язык разметки (XML) 1.0 (пятое издание) Рекомендация W3C 26 ноября 2008 г.[электронное издание] – режим доступа: https://www.w3.org/TR/2008/REC-xml-20081126/

27. Частное земледелие. // Прянишников Д. Н. - Типо-литография В. Рихтер, 1901 – 407 с.

28. Культурные растения и их сородичи. Систематика, география, цитогенетика, иммунитет, экология, происхождение, использование. // Жуковский П.М. - Ленинград, «Колос», 1971 – 752 с.

29. Растениеводство / Г.С. Посыпанов, В.Е. Долгодворов, Б.Х. Жеруков [и др.] ; под ред. Г.С. Посыпанова. — М. : ИНФРА-М, 2019. — 612 с.

30. Classification of crops - APPENDIX 3// FAO, World Programme for the Census of Agriculture 2010 (Всемирная программа сельскохозяйственной переписи 2010 года), 2010 – 142-146с.

31. ОКПД 2 — Общероссийский классификатор продукции по видам экономической деятельности [Электронный ресурс]: Классификатор ОК 034-2014 (КПЕС 2008), ответственный – Росстат, введен по Приказу Росстандарта от 31.01.2014 №14-ст, начал действовать с 01.02.2014 // Справочная система «Консультант-плюс» - Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_163703/

32. Государственный реестр селекционных достижений, допущенных к использованию. Т.1. «Сорта растений» (официальное издание). // Государственная комиссия РФ по испытанию и охране селекционных достижений, М.:ФГБНУ «Росинформагротех», 2018. – 504 с.

33. SPARQL 1.1 Query Language W3C Recommendation 21 March 2013 [электронный ресурс] – режим доступа: https://www.w3.org/TR/sparql11-query/

34. SPARQL 1.1 Overview W3C Recommendation 21 March 2013 [электронный ресурс] – режим доступа: https://www.w3.org/TR/sparql11-overview/

35. Protégé User Documentation [электронный ресурс] – режим доступа: https://protegewiki.stanford.edu/wiki/DLQueryTab

36. A Survey on Automatic Mapping of Ontology to Relational Database Schema // Afzal Humaira, Naz Tabbasum and Sadiq Ayesha - Research Journal of Recent Sciences - Vol. 4(4), April (2015), с. 66-7

37. Mapping between the OBO and OWL ontology languages // Syed Hamid Tirmizi, Stuart Aitken, Dilvan A Moreira, Chris Mungall, Juan Sequeda, Nigam H Shah, and Daniel P Miranker - J Biomed Semantics. 2011; 2(Suppl 1): S3.

38. OWLMap: Fully Automatic Mapping of Ontology into Relational Database Schema // Humaira Afzal, Mahwish Waqas, Dr. Tabbassum Naz – International Journal of Advanced Computer Science and Applications, Vol. 7, No. 11, 2016 – с. 7 – 15

39. Конвертер онтологической базы знаний станков и деталей машиностроительного производства в объектную базу данных на основе ANTLR // В.А. Игруша, С.С. Сосинская – Онтология проектирования. – 2016. – Т.6, №3(21) – с.278-286

40. Script for converting an OWL file to the DB backend // официальное сообщество поддержки Protege [электронный ресурс]: режим доступа - https://protegewiki.stanford.edu/wiki/ConvertToDBScript

ПРИЛОЖЕНИЕ А

Код скрипта для интеграции онтологии в базу данных

```
import xml.dom.minidom
import pymysql
import mysql.connector
import pymysql.cursors

dom = xml.dom.minidom.parse('primer.xml')
pymysql.install_as_MySQLdb()

def handleClass(text):
    classes = []
    cl = []
    cl = text.getElementsByTagName('Class')
    for elem in cl:
        fixed = elem.attributes['IRI'].value
        l1 = fixed.find('#')
        fixed = fixed[l1+1:]
        #text = fixed[0]
       # print(text[text.find('#')+1:])
        classes.append(fixed)
        #print(fixed)
    listcl=list(set(classes))
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     # charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for el in listcl:
                #print(el)
                sql = "INSERT IGNORE INTO classes (name) VALUES(%s);"
                cursor.execute(sql,el)
                connection.commit()
    finally:
        connection.close()

def handleIndividuals(text1):
    items = []
    ind = []
    ind = text1.getElementsByTagName('NamedIndividual')
    for element in ind:
        fixed1 = element.getAttribute('IRI')
        l1 = fixed1.find('#')
        # l2 = fixed.find('#')
        fixed1 = fixed1[l1+1:]
        # fixed = fixed[l2+1:]
        items.append(fixed1)
        #print(fixed)
    listcl=list(set(items))
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',,
                                     cursorclass=pymysql.cursors.DictCursor)
```

Продолжение приложения А

```
        with connection.cursor() as cursor:
            for el in listcl:
                #print(el)
                sql = "INSERT IGNORE INTO individuals (name_individuals) VAL-
UES(%s);"
                cursor.execute(sql,el)
                connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleDataproperty(text2):
    items = []
    ind = []
    ind = text2.getElementsByTagName('DataProperty')
    for element in ind:
        fixed = element.getAttribute('IRI')
        l1 = fixed.find('#')
        # l2 = fixed.find('#')
        fixed = fixed[l1+1:]
        # fixed = fixed[l2+1:]
        items.append(fixed)
        #print(fixed)
    listcl=list(set(items))
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for el in listcl:
                #print(el)
                sql = "INSERT IGNORE INTO data_property (name_data_property)
VALUES(%s);"
                cursor.execute(sql,el)
                connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleObjectproperty(text3):
    items = [] #все элементы object_property
    domain_classes = [] #object_property содержащие свойство domain
    range_classes = [] #object_property содержащие свойство range
    ind = []
    ind = text3.getElementsByTagName('ObjectProperty')
    for element in ind:
        fixed = element.getAttribute('IRI')
        l1 = fixed.find('#')
        fixed = fixed[l1+1:]
        items.append(fixed)
    listcl=list(set(items))
    ind = text3.getElementsByTagName('ObjectPropertyDomain')
    for element in ind:
```

Продолжение приложения А

```
        nodelist = element.childNodes
        dom_s = nodelist.item(1).getAttribute('IRI')
        l2 = dom_s.find('#')
        where_add_domain = dom_s[l2+1:] # какому свойству добавляем домен
        domain_classes.append(where_add_domain)
        #print(where_add_domain)
        dom_s2 = nodelist.item(3)
        if (dom_s2.getAttribute('IRI')):
            class_domain = dom_s2.getAttribute('IRI')
            l1 = class_domain.find('#')
            class_domain = class_domain[l1+1:] # какой класс добавляем в домен
            domain_classes.append(class_domain)
            #print(class_domain)
        else:
            dom_s3 = nodelist.item(3)
            dom_s4 = dom_s3.childNodes #разбор тега ObjectSomeValuesFrom анало-
гичен ObjectPropertyDomain, см.выше
            dom_s5 = dom_s4.item(1).getAttribute('IRI') #object property
            l3 = dom_s5.find('#')
            class_domain = dom_s5[l2+1:]
            dom_s6 = dom_s4.item(3).getAttribute('IRI') # some class
            l4 = dom_s6.find('#')
            class_domain = class_domain + ' some ' + dom_s6[l2+1:] #obj_prop
some class
            domain_classes.append(class_domain)
    ind = text3.getElementsByTagName('ObjectPropertyRange')
    for element in ind:
        nodelist = element.childNodes
        dom_s = nodelist.item(1).getAttribute('IRI')
        l2 = dom_s.find('#')
        where_add_range = dom_s[l2+1:] # какому свойству добавляем домен
        range_classes.append(where_add_range)
        #print(where_add_domain)
        dom_s2 = nodelist.item(3)
        if (dom_s2.getAttribute('IRI')):
            class_range = dom_s2.getAttribute('IRI')
            l1 = class_range.find('#')
            class_range = class_range[l1+1:] # какой класс добавляем в домен
            range_classes.append(class_range)
            #print(class_domain)
        else:
            dom_s3 = nodelist.item(3)
            dom_s4 = dom_s3.childNodes #разбор тега ObjectSomeValuesFrom анало-
гичен ObjectPropertyDomain, см.выше
            dom_s5 = dom_s4.item(1).getAttribute('IRI') #object property
            l3 = dom_s5.find('#')
            class_range = dom_s5[l2+1:]
            dom_s6 = dom_s4.item(3).getAttribute('IRI') # some class
            l4 = dom_s6.find('#')
            class_range = class_range + ' some ' + dom_s6[l2+1:] #obj_prop some
class
            range_classes.append(class_range)
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     cursorclass=pymysql.cursors.DictCursor)
```

Продолжение приложения А

```
        with connection.cursor() as cursor:
            for el in listcl:
                print(el)
                sql = "INSERT IGNORE INTO object_property (name_object_property)
VALUES(%s);"
                cursor.execute(sql,el)
                for i in range(0,len(domain_classes),2):
                    e = domain_classes[i]
                    if(el == e):
                        sql = "UPDATE object_property SET domain = %s WHERE
name_object_property = %s;"
                        some = ( domain_classes[i+1], el)
                        cursor.execute(sql,some)
                        connection.commit()
                        continue
                for i in range(0,len(range_classes),2):
                    e = range_classes[i]
                    if(el == e):
                        sql = "UPDATE object_property SET range_prop = %s WHERE
name_object_property =%s;"
                        some = (range_classes[i+1],el)
                        cursor.execute(sql,some)
                        connection.commit()
                        continue
                connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleSubclass(text4):
    items = [] #все значения из тега subclassof
    ind = []
    ind = text4.getElementsByTagName('SubClassOf')
    for element in ind:
        nodelist = element.childNodes
        subcl = nodelist.item(1).getAttribute('IRI') #подкласс
        l1 = subcl.find('#') #так как онтологии соединены, перед названием стоит
пространство имен
        subcl = subcl[l1+1:]  # надо его убрать
        items.append(subcl)
        supercl = nodelist.item(3).getAttribute('IRI')
        l2 = supercl.find('#')
        supercl = supercl[l2+1:]
        items.append(supercl)
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     # charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for i in range(0,len(items),2): #цикл for с шагом 2, так как в item
подкласс+надкласс
                    e = items[i]
                    check_sql = "SELECT * FROM subclass WHERE
(name_subclass,name_superclass) \
```

Продолжение приложения А

```
                                in (SELECT name_subclass, name_superclass FROM
subclass group by name_subclass \
                              , name_superclass having count(*)>1);" #проверка на
существование строки
                    if (cursor.execute(check_sql)>0):
                        continue
                    sql = "INSERT INTO subclass (name_subclass, name_superclass)
VALUES(%s,%s);"
                    some = (e, items[i+1]) #набор переменных
                    cursor.execute(sql,some) #исполнение запроса с несколькими
переменными
                    connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleClassIndividuals(text4):
    items = [] #все значения из тега classassertion
    ind = []
    ind = text4.getElementsByTagName('ClassAssertion')
    for element in ind:
        nodelist = element.childNodes
        class_name = nodelist.item(1).getAttribute('IRI') #класс
        l1 = class_name.find('#') #так как онтологии соединены, перед названием
стоит пространство имен
        class_name = class_name[l1+1:]  # надо его убрать
        items.append(class_name)
        individual_name = nodelist.item(3).getAttribute('IRI') #экземпляр
        l2 = individual_name.find('#')
        individual_name = individual_name[l2+1:]
        items.append(individual_name)
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     # charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for i in range(0,len(items),2): #цикл for с шагом 2, так как в item
класс+экземпляр
                    e = items[i]
                    check_sql = "SELECT * FROM class_individuals WHERE
(name_class,name_individuals) \
                              in (SELECT name_class, name_individuals FROM
class_individuals group by \
                              name_class , name_individuals having count(*)>1);"
#проверка на существование строки
                    if (cursor.execute(check_sql)>0):
                        continue
                    sql = "INSERT INTO class_individuals (name_class,
name_individuals) VALUES(%s,%s);"
                    some = (e, items[i+1]) #набор переменных
                    cursor.execute(sql,some) #исполнение запроса с несколькими
переменными
                    connection.commit()
    except mysql.connector.Error as error :
```

Продолжение приложения А

```
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleObjectPropertyRelate(text5):
    items = [] #все значения из тега object property assertion
    ind = []
    ind = text5.getElementsByTagName('ObjectPropertyAssertion')
    for element in ind:
        nodelist = element.childNodes
        obj_name = nodelist.item(1).getAttribute('IRI') #object property
        l1 = obj_name.find('#') #так как онтологии соединены, перед названием
стоит пространство имен
        obj_name = obj_name[l1+1:]  # надо его убрать
        items.append(obj_name)
        individual_name1 = nodelist.item(3).getAttribute('IRI') #individuals 1
        l2 = individual_name1.find('#')
        individual_name1 = individual_name1[l2+1:]
        items.append(individual_name1)
        individual_name2 = nodelist.item(5).getAttribute('IRI') #individuals 2
        l3 = individual_name2.find('#')
        individual_name2 = individual_name2[l3+1:]
        #print(obj_name)
        items.append(individual_name2)
    ind2 = text5.getElementsByTagName('NegativeObjectPropertyAssertion')
    for element in ind2:
        nodelist = element.childNodes
        obj_name = nodelist.item(1).getAttribute('IRI') #object property
        l1 = obj_name.find('#') #так как онтологии соединены, перед названием
стоит пространство имен
        obj_name = 'not ' + obj_name[l1+1:]  # надо его убрать
        items.append(obj_name)
        individual_name1 = nodelist.item(3).getAttribute('IRI') #individuals 1
        l2 = individual_name1.find('#')
        individual_name1 = individual_name1[l2+1:]
        items.append(individual_name1)
        individual_name2 = nodelist.item(5).getAttribute('IRI') #individuals 2
        l3 = individual_name2.find('#')
        individual_name2 = individual_name2[l3+1:]
        #print(obj_name)
        items.append(individual_name2)
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     # charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for i in range(0,len(items),3): #цикл for с шагом 2, так как в item
класс+экземпляр
                e = items[i]
                check_sql = "SELECT * FROM objprop_relate WHERE
(name_object_property,name_individual_1\
                    , name_individual_2) in (SELECT
name_object_property,name_individual_1, name_individual_2 \
```

Продолжение приложения А

```
                    FROM objprop_relate group by  name_object_property,\
                    name_individual_1,name_individual_2 having count(*)>1);"
#проверка на существование строки
                if (cursor.execute(check_sql)>0):
                    continue
                sql = "INSERT INTO objprop_relate
(name_object_property,name_individual_1, name_individual_2) \
                    VALUES(%s,%s,%s);"
                some = (e, items[i+1], items[i+2]) #набор переменных
                cursor.execute(sql,some) #исполнение запроса с несколькими
переменными
                connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()

def handleDataPropertyRelate(text4):
    items = [] #все значения из тега data propertyassertion
    ind = []
    ind = text4.getElementsByTagName('DataPropertyAssertion')
    for element in ind:
        nodelist = element.childNodes
        #print(nodelist)
        ie =1
        first = nodelist.item(ie).tagName
        #print(first)
        if (first == 'Annotation'):
            ie = 3
        data_property_name = nodelist.item(ie).getAttribute('IRI') #класс
        l1 = data_property_name.find('#') #так как онтологии соединены, перед
названием стоит пространство имен
        data_property_name = data_property_name[l1+1:]  # надо его убрать
        items.append(data_property_name)
        individual_name = nodelist.item(ie+2).getAttribute('IRI') #экземпляр
        l2 = individual_name.find('#')
        individual_name = individual_name[l2+1:]
        items.append(individual_name)
        datatype = nodelist.item(ie+4).getAttribute('datatypeIRI') #экземпляр
        l3 = datatype.find('#')
        datatype = datatype[l3+1:]
        items.append(datatype)
        value_data = nodelist.item(ie+4).firstChild #экземпляр
        if (value_data == None):
            items.append(value_data)
        else:
            value_data = value_data.nodeValue
            items.append(value_data)
    try:
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     # charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for i in range(0,len(items),4): #цикл for с шагом 4, так как в item
класс+экземпляр
```

Окончание приложения А

```
                    e = items[i]
                    check_sql = "SELECT * FROM dataprop_relate WHERE
(name_data_property,name_individual,\
                    datatype, value_data) in (SELECT
name_data_property,name_individual, datatype,\
                    value_data  FROM dataprop_relate group by
name_data_property,name_individual,\
                    datatype,value_data having count(*)>1);" #проверка на суще-
ствование строки
                    if (cursor.execute(check_sql)>0):
                        continue
                    sql = "INSERT INTO dataprop_relate
(name_data_property,name_individual,datatype,\
                    value_data) VALUES(%s,%s, %s, %s);"
                    some = (e, items[i+1], items[i+2], items[i+3]) #набор пере-
менных
                    cursor.execute(sql,some) #исполнение запроса с несколькими
переменными
                    connection.commit()
    except mysql.connector.Error as error :
        connection.rollback()
        print("Failed to insert into MySQL table {}".format(error))
    finally:
        connection.close()
handleClass(dom)
handleIndividuals(dom)
handleDataproperty(dom)
handleObjectproperty(dom)
handleSubclass(dom)
handleClassIndividuals(dom)
handleObjectPropertyRelate(dom)
handleDataPropertyRelate(dom)
```

ПРИЛОЖЕНИЕ Б

Результат проверки в системе «АНТИПЛАГИАТ»

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Сибирский федеральный университет»

НАУЧНАЯ БИБЛИОТЕКА

660049, Красноярск, пр. Свободный ,79/10, тел.(3912) 2-912-820, факс (3912) 2-912-773
E-mail: bik@sfu-kras.ru

ОТЧЕТ
о результатах проверки в системе «АНТИПЛАГИАТ»

Автор: Кузнецова Анастасия Алексеевна
Заглавие: диплом
Вид документа: Выпускная квалификационная работа бакалавра

Итоговая оценка оригинальности: 84,6%

Подготовлено автоматически с помощью системы «Антиплагиат»
дата: 16.06.2019

ПРИЛОЖЕНИЕ В

Плакаты презентации

Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
кафедра систем искусственного интеллекта

БАКАЛАВРСКАЯ РАБОТА

09.03.02.05 - «Информационные системы и технологии в административном управлении»

Разработка онтологии оценивания земель сельскохозяйственного назначения

Руководитель канд. техн. наук, доцент К.В. Раевич
Студент гр. КИ15-12Б А.А. Кузнецова

Красноярск 2019

1

Актуальность

- В 2017 году в Правительстве Красноярского края была сформирована проектная инициатива «Внедрение технологий дистанционного зондирования Земли в целях повышения эффективности государственного управления».
- По одному из соглашений, подписанных на Петербургском международном экономическом форуме - 2019, Красноярский край станет пилотной площадкой по внедрению технологий дистанционного зондирования Земли и цифровизации в государственном управлении.

2

Цель и задачи:

Цель:

создание онтологии оценивания ЗСХН и интегрирование в систему космического агромониторинга

Задачи:

- обзор предметной области;
- формирование терминологической части и выстраивание иерархии;
- разработка онтологии предметной области в среде Protégé и интеграция онтологии в СУБД.

3

Анализ предметной области (1)

- Ontological Foundations for Geographic Information Science - David Mark - University at Buffalo, 2003
- A Survey of Qualitative Spatial Representations - Chen, J - Jilin University, 2013
- Colect Earth Danae Maniatis – University of Oxford (2016)
- Earth observation metadata - Xiaolei Wang - Wuhan University (2016)
- Ontological foundations for feature-based modeling
- - Emilio M.Sanfilippo - Laboratoire des Sciences du Numérique de Nantes
- (2018)
- Разработка и реализация формальных онтологий пространственных данных и сервисов - Динь Ле Дат - МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ – 2008
- Обработка пространственной информации - И.Н.Розенберг – АО НИИАС (2015)
- ГИС сегодня: состояние, перспективы, решения - Ю. И. Шокин, В. П. Потапов – НТУ, КФ ИВТ СО РАН (2015)
- Geoknowledge - В.Я. Цветков – МИРЭА -2016
- Онтологии информационного моделирования – В.Я. Цветков -2018

4

Анализ предметной области (2)

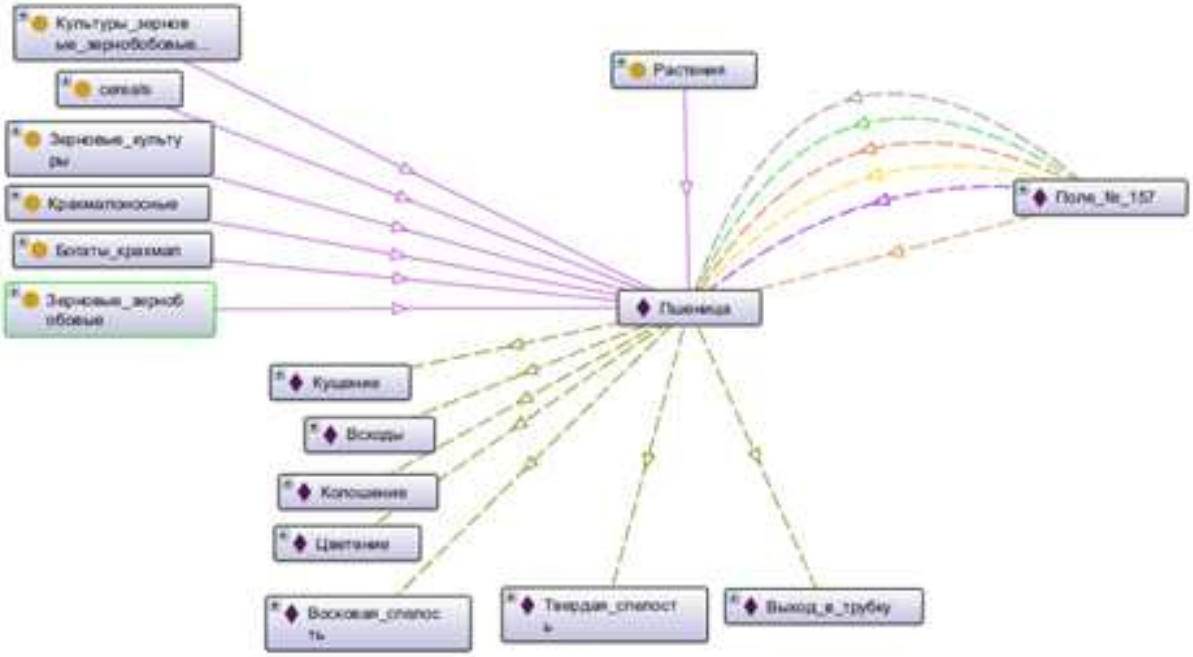
- Классификации почв – В.В. Докучаев (СПбГУ), А.В. Якушев (МГУ)
- Классификации растений – Прянишников, Жуковский (РГАУ МСХА), FAO (Продовольственная организация ООН), ОКПД 2 (Общероссийский Классификатор продукции)
- Атмосферные явления – открытый архив погоды (rp5, Gismeteo)

5

Структура онтологии, верхний уровень

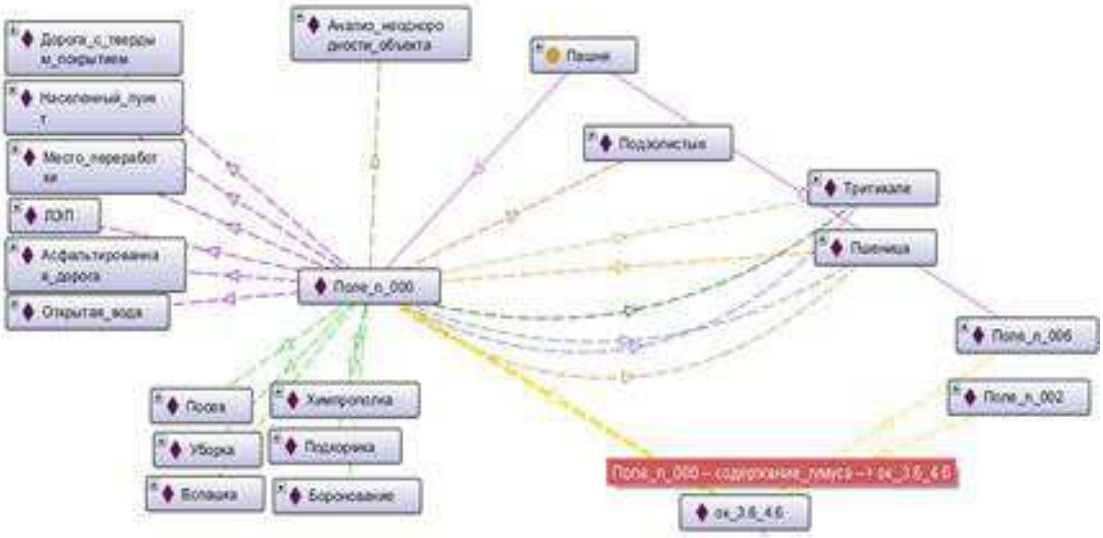
6

Описание культуры (граф)



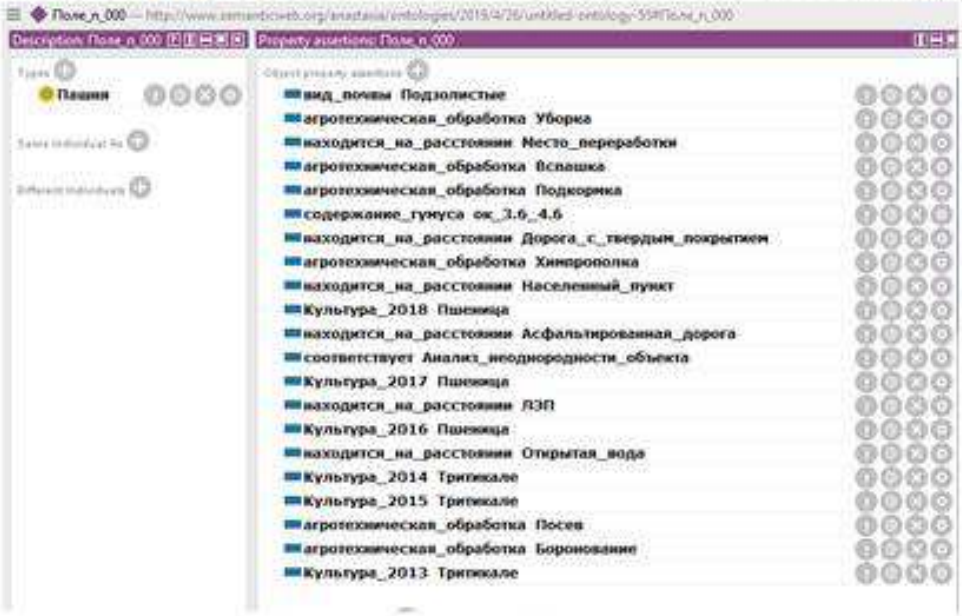
7

Описание ЗСХН (граф)



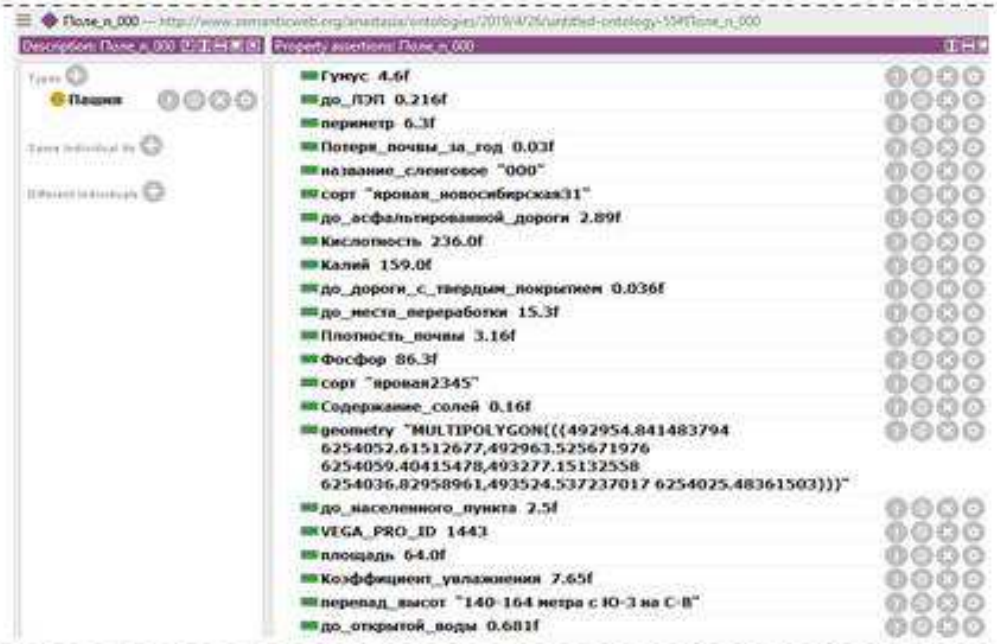
8

Объектные свойства ЗСХН



9

Свойства данных ЗСХН



10

Запрос SPARQL

```
PREFIX значение: <IRI>
SELECT [DISTINCT] [REDUCED]?переменная(столбец) / *(все значения)
   WHERE {
            ?субъект свойство ?объект.
            [ VALUES ?объект {} ]
            [ OPTIONAL { ?субъект свойство ?объект} ]
            [ FILTER [ NOT EXIST {} ] [ regex(?объект, "маска/значение") ] ?объект условие ]
            [ BIND ( [IF ()] [CONCAT () ] AS ?объект_ ]
            [ UNION {} ]
            [ MINUS {} ]
   }
   [ ORDER BY ]
   [ DESC (?объект) ]
   [ OFFSET количество ]
   [ HAVING () ]
```

Где PREFIX – пространство имен,
DISTINCT – исключение повторов,
REDUCED – выдача всех решений,
OPTIONAL – выдача решений с незаполненными ключевыми полями,
FILTER – фильтр по существованию/ регулярным выражениям/ условиям,
BIND – связанные запросы,
CONCAT – объединить с графами,
UNION – вложенные запросы,
MINUS – исключить определенные решения,
ORDER BY – сортировка,
DESC – по убыванию,
OFFSET – исключить n решений

11

Выдача запроса SPARQL

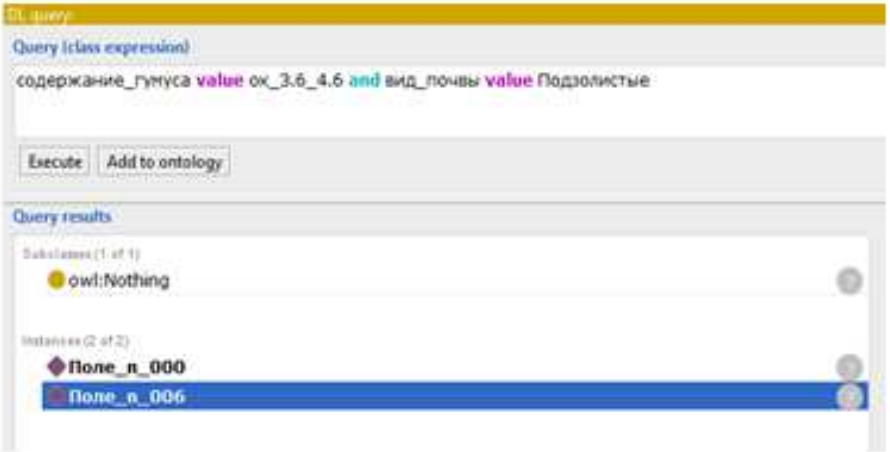
SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ont: <http://www.semanticweb.org/anastasia/ontologies/2019/4/26/untitled-ontology-55#>
SELECT distinct ?individual ?number
WHERE {
            ?individual rdf:type owl:NamedIndividual.
            ?individual rdf:type ont:Пашня.
            ?individual ont:агротехническая_обработка ont:Подкормка.
            ?individual ont:Гумус ?number . FILTER (?number<5.0)
}
```

individual	number
Поле_n_000	"4.6"^^<http://www.w3.org/2001/XMLSchema#float>

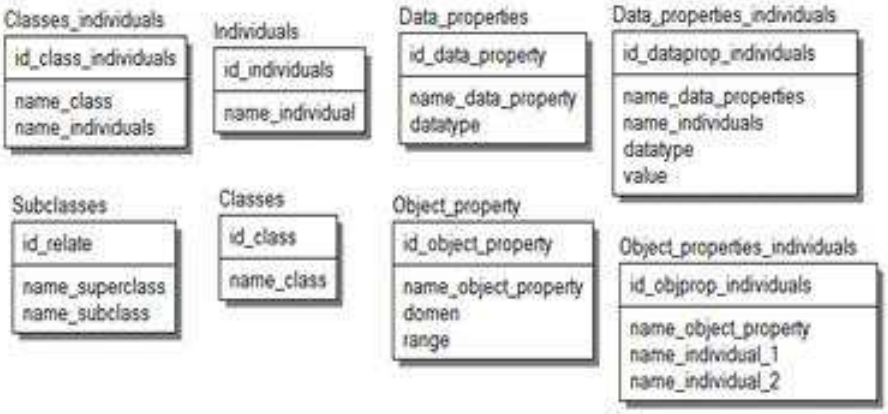
12

Выдача запроса DL query



13

Физическая модель базы данных для отображения онтологии



14

Основная логика скрипта (python) отображения онтологии в базу данных

```
def handleClass(text):
    classes = []
    cl = []
    cl = text.getElementsByTagName('Class') #находим тег по названию
    for elem in cl:
        fixed = elem.attributes['IRI'].value #находим атрибут «IRI» -название
        ll = fixed.find('#')
        fixed = fixed[ll+1:] #обрезаем название- в Protégé в названиях ссылки
        classes.append(fixed)
        listcl=list(set(classes))
    try: #попытка подключения к бд и вставка найденных данных(классов)
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='12345',
                                     db='protege',
                                     cursorclass=pymysql.cursors.DictCursor)
        with connection.cursor() as cursor:
            for el in listcl:
                sql = "INSERT IGNORE INTO classes (name) VALUES(%s);"
                cursor.execute(sql,el)
                connection.commit()
    finally:
        connection.close()
```

▶ 15

Фрагмент таблицы в MySQL (отображение онтологии)

```
| 735 | Сумма_осадков_за_год                                          |
| 661 | Сумма_положительных_температур_за_вегетационный_период        |
| 784 | Схемы_чередования_культур_в_севооборотах                      |
| 788 | Табак_необработанный                                          |
| 773 | Тепловые_свойства                                             |
| 643 | Теплоемкость                                                  |
| 541 | Теплопроводность                                              |
| 636 | Технические                                                   |
| 768 | Технические_культуры                                          |
| 786 | Травы_многолетние                                             |
| 544 | Травы_однолетние                                              |
| 686 | Трансформация_угодий                                          |
| 673 | Тропические_субтропические_фрукты                             |
| 696 | Тростник_сахарный                                             |
| 745 | Тыквенные_Овощи                                               |
| 674 | Увлажнение                                                    |
| 686 | Удельный_вес_почвы                                            |
| 757 | Уклон_рельефа                                                 |
| 645 | Уплотненность                                                 |
| 650 | Фактор_формы                                                  |
| 548 | Фенологическая_фаза                                           |
| 689 | Физические_свойства_почвы                                     |
| 638 | Физические_факторы_рельефа                                    |
| 742 | Форма_поля                                                    |
| 589 | Форма_рельефа                                                 |
| 787 | Функциональные_свойства                                       |
| 765 | Химический                                                    |
| 583 | Цветочно-декоративные                                         |
| 682 | Цветы_срезанные_кустовые                                      |
| 752 | Цитрусовые_субтропические                                     |
| 684 | Щебнистости_каменистости                                      |
| 670 | Щелочность                                                    |
| 612 | Эродированность                                               |
| 748 | Эфиронасличные                                                |
| 617 | Эффективное_расстояние_до_объектов_инфраструктуры             |
| 774 | Ягодники                                                      |
| 572 | Ягодные                                                       |
| 738 | Ягоды_плоды                                                   |
+-----+---------------------------------------------------------------+
271 rows in set (0.89 sec)

mysql>
```

▶ 16

Представление работы (1)


ПРОСПЕКТ СВОБОДНЫЙ

Международная студенческая конференция «ПРОСПЕКТ СВОБОДНЫЙ - 2019».

Секция «Системный анализ, управление и программная инженерия»

Институт космических и информационных технологий

СЕРТИФИКАТ очного участия в конференции

Выдан Кемчерской Анастасии Алексеевне

участнику Международной конференции студентов, аспирантов и молодых ученых «Проспект Свободный - 2019», состоявшейся 22-26 апреля 2019 года, г. Красноярск.

Председатель С.В. Чебыкин

Секретарь Н.Б. Прокопчина

22.04.2019 г.

ПРОСПЕКТ СВОБОДНЫЙ 2019

17

Представление работы (2)

Очное выступление на круглом столе «Возможности использования данных ДЗЗ для распознавания почвенно-растительного покрова».

24 мая 2019 года.

СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

www.kgau.ru

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
INSTITUTE OF SPACE AND INFORMATION TECHNOLOGY

18

Заключение

- Выполнен обзор предметной области: проанализированы способы представления знаний и методы построения ГИС на онтологиях;
- сформирована терминологическая часть: свойства ЗСХН разбиты на 6 групп, каждая описана;
- разработана онтология предметной области в среде Protégé;
- проведена интеграция онтологии в СУБД MySQL с помощью программы на языке программирования Python.

19

Спасибо за внимание!

20

ПРИЛОЖЕНИЕ Г

Сертификат об участии в конференции

ПРОСПЕКТ СВОБОДНЫЙ

ПРОСПЕКТ СВОБОДНЫЙ
МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ МОЛОДЫХ УЧЕНЫХ

Международная студенческая конференция «ПРОСПЕКТ СВОБОДН
посвященная Международному году Периодической таблицы химических элементов.

Секция «Системный анализ, управление и программная ин

Институт космических и информационных технологий

СЕРТИФИКАТ очного уч
в конфер

Выдан Кузнецовой Анастасии Алексеевне
(фамилия, имя, отчество полностью)

участнику Международной конференции студентов, аспирантов и молод
«Проспект Свободный – 2019», состоявшейся 22–26 апреля 2019 года, г.

Председатель

Секретарь

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕН
ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра систем искусственного интеллекта

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Г.М. Цибульский
подпись
«25» 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – Информационные системы и технологии

Разработка онтологии для оценивания
земель сельскохозяйственного назначения

Руководитель _____ доц., канд. техн. наук К.В. Раевич
подпись, дата

Выпускник _____ 25.06.19 А.А. Кузнецова
подпись, дата

Красноярск 2019