

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра Систем информатики

Направление подготовки 09.06.01 – Информатика и вычислительная техника

**ОТЧЕТ**

**Обучающегося Козубенко Андрея Алексеевича группы № 22215 курса 4**  
(Ф.И.О. полностью)

**Тема задания:** Разработка репозитория для редактирования онтологии предметной области

Новосибирск 2025

## Оглавление

Введение	3
Реализация репозитория онтологий	4
Листинги	5
Заключение	6
Список литературы (если будет)	7

## Введение

Целью данной работы является разработка Python-репозитория для работы с онтологиями предметной области, хранящимися в графовой базе данных Neo4j.

Для достижения цели были поставлены следующие задачи:

- реализовать методы для управления классами онтологии (создание, обновление, удаление, получение родителей и потомков);
- реализовать методы для работы с объектами классов (создание, обновление, удаление);
- реализовать поддержку атрибутов классов: `DatatypeProperty` и `ObjectProperty`;
- реализовать сбор сигнатуры класса (`collect_signature`), включающей все его свойства;
- реализовать метод получения всей онтологии и корневых классов.

## Реализация репозитория онтологий

Для реализации лабораторной был создан класс **OntologyRepository**, наследующийся от **Neo4jRepository**. В нем были реализованы следующие группы методов:

### 1. Работа с онтологией:

- `get_ontology()` – получение всей онтологии;
- `get_ontology_parent_classes()` – поиск корневых классов;
- `get_class()`, `get_class_parents()`, `get_class_children()` – работа с отдельным классом.

### 2. Управление классами:

- `create_class()`, `update_class()`, `delete_class()` – создание, редактирование и удаление классов вместе с потомками и атрибутами.

### 3. Атрибуты классов:

- `add_class_attribute()` / `delete_class_attribute()` – работа с `DatatypeProperty`;
- `add_class_object_attribute()` / `delete_class_object_attribute()` – работа с `ObjectProperty`.

### 4. Объекты классов:

- `create_object()`, `update_object()`, `get_object()`, `delete_object()`.

### 5. Сигнатуры:

- `collect_signature()` – сбор информации обо всех параметрах (`DatatypeProperty`) и объектных связях (`ObjectProperty`).

## Тестирование и пример использования

Пример сценария (см. `example_usage_ontology.py`)

### Реализация

<https://github.com/andrew-kozubenko/Linguistics>

Находится в `lab2/ontology_repository.py`

## Листинги

Листинг 1 – Метод для сбора сигнатуры класса:

```
def collect_signature(self, class_uri: str) -> Dict[str, Any]:
    """
    Сбор сигнатуры класса: все DatatypeProperty и ObjectProperty.
    Возвращает словарь:
    {
        params: [{title, uri}, ...],
        obj_params: [{title, uri, target_class_uri, relation_direction}, ...]
    }
    relation_direction:
    1 - класс <-[:DOMAIN]- ObjectProperty
    -1 - ObjectProperty -[:RANGE]-> класс
    """
    cypher = """
    MATCH (c:Class {uri: $uri})
    OPTIONAL MATCH (c)-[:DOMAIN]-(dp:DatatypeProperty)
    OPTIONAL MATCH (c)-[:DOMAIN]-(op1:ObjectProperty)-[:RANGE]->(rc1:Class)
    OPTIONAL MATCH (c)-[:RANGE]-(op2:ObjectProperty)-[:DOMAIN]->(rc2:Class)

    RETURN
        collect(DISTINCT dp) AS datatype_props,
        collect(DISTINCT {prop: op1, target: rc1, direction: 1}) AS obj_props_pos,
        collect(DISTINCT {prop: op2, target: rc2, direction: -1}) AS obj_props_neg
    """
```

Данный метод позволяет собрать все параметры и связи класса, что делает возможным динамическое построение объектов.

Листинг 2 – Метод для удаления класса вместе с его потомками, объектами и атрибутами:

```
1 usage
def delete_class(self, class_uri: str) -> bool:
    """
    Удаляет класс вместе со всеми потомками, их объектами и атрибутами.
    """
    # 1. Находим все классы: целевой и его потомков
    cypher_classes = """
    MATCH (c:Class {uri: $uri})
    OPTIONAL MATCH (c)-[:SUBCLASS_OF*0..]- (descendant:Class)
    WITH collect(DISTINCT c) + collect(DISTINCT descendant) AS classes_to_delete
    UNWIND classes_to_delete AS cls
    RETURN cls.uri AS uri
    """
```

```

# 2. Находим все объекты этих классов
cypher_objects = """
MATCH (o:Object)
WHERE o.class_uri IN $classes
RETURN collect(o.uri) AS objects_to_delete
"""

res_objects = self.run_custom_query(cypher_objects, params: {"classes": classes_uris})
object_uris = res_objects[0].get("objects_to_delete", []) if res_objects else []

# 3. Находим все DatatypeProperty и ObjectProperty этих классов
cypher_props = """
MATCH (p)
WHERE (p:DatatypeProperty OR p:ObjectProperty)
    AND EXISTS {
        MATCH (p)-[:DOMAIN]->(c:Class)
        WHERE c.uri IN $classes
    }
RETURN collect(p.uri) AS props_to_delete
"""

```

Данный метод позволяет удалить класс вместе с его потомками, объектами и атрибутами.

## Заключение

В результате проведенной работы были реализованы:

1. Методы для работы с классами и объектами онтологии.
2. Методы для добавления и удаления свойств классов (DatatypeProperty, ObjectProperty).
3. Метод сбора сигнатуры класса для использования при создании и обновлении объектов.
4. Полный набор операций для редактирования онтологии в Neo4j.

Таким образом, все задачи, поставленные во введении, были выполнены.

## Список литературы

1. Neo4j documentation: <https://neo4j.com/docs>