Основы теории систем

**Отчет по лабораторной №3**

«Формирование онтологической модели с помощью Protege и Neo4j»

Вариант 1

Выполнил:

Студент 1 курса   
Группы ИВТИИбд-12  
Абанин Михаил

Ульяновск, 2025

**Задание:**

1. Сформировать онтологическую модель структуры университета.

2. Реализовать эту модель в Protege и Neo4j.

3. Произвести сравнительный анализ двух инструментов.

4. Сделать заключение о Protege и Neo4j.

Требования:

* минимум 3 класса
* у каждого класса – хотя бы 1 свойство
* минимум 2 типа связи
* минимум 2 экземпляра каждого класса
* минимум 2 правила логического вывода

**Классы:**

University - Университет

Faculty - Факультет

Department - Кафедра

Teacher(Professor) - Преподаватель

Student - Студент

Course — Курс

**Свойства:**

Имя(Название)

**Экземпляры:**

**University:**

Университет

**Faculty:**

Факультет информационных систем и технологий

Строительный факультет

**Department:**

Архитектурно-строительное проектирование

Вычислительная техника

**Professor(Teacher):**

Виталий Иванович Тур

Лапшов Юрий Александрович

**Student:**

Михаил

Александр

**Course:**

ИВТИИбд-12

ПГСбд-11

**Связи:**

Включает

IS\_PART\_OF

WORKS\_FOR

STUDIES\_AT

BELONGS\_TO

TEACHES

ENROLLED\_IN

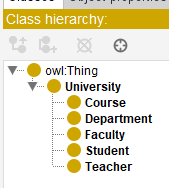
**Правила логического вывода:**

Если студент записан на курс, который ведет преподаватель → студент имеет этого преподавателя

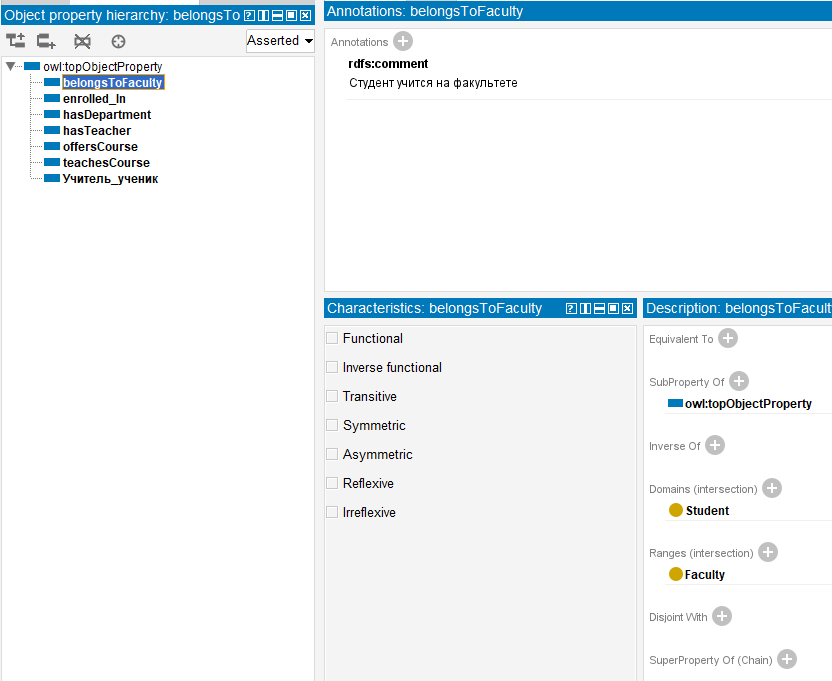
Если преподаватель работает на кафедре, которая принадлежит факультету → преподаватель принадлежит этому факультету

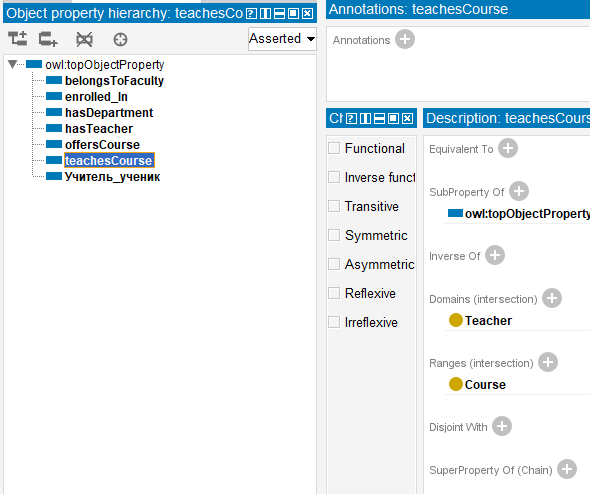
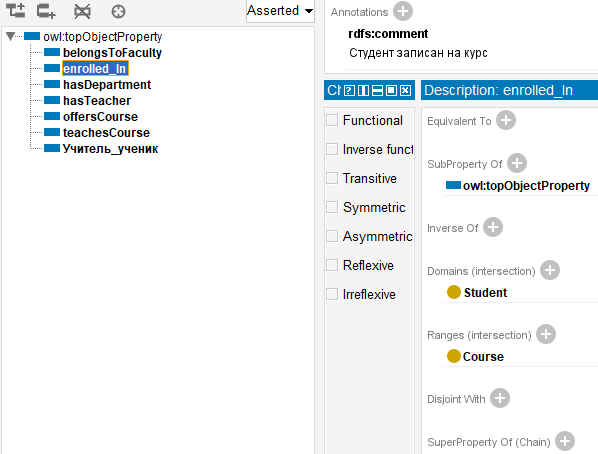
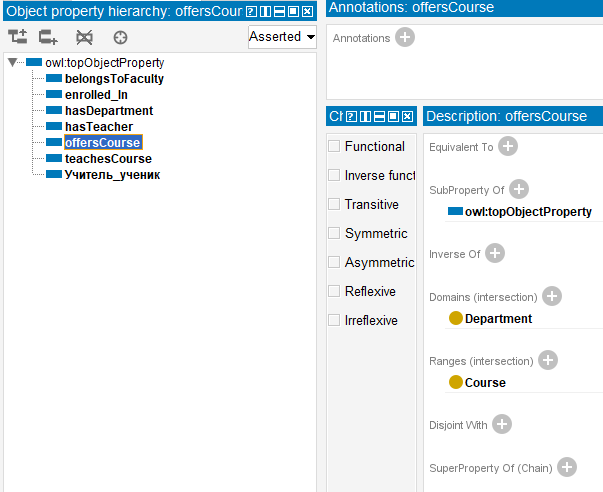
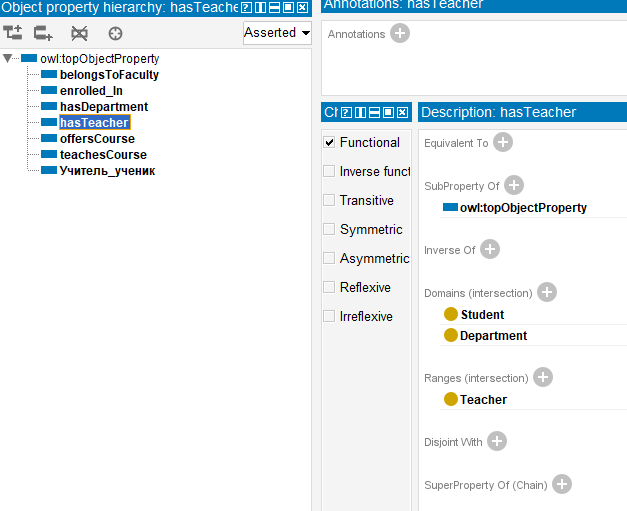
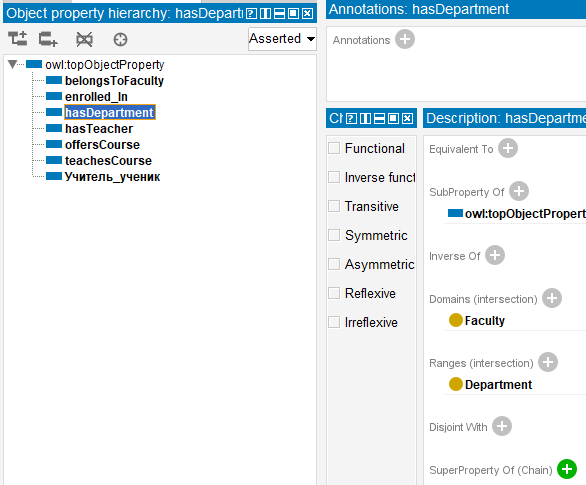
**Protege**

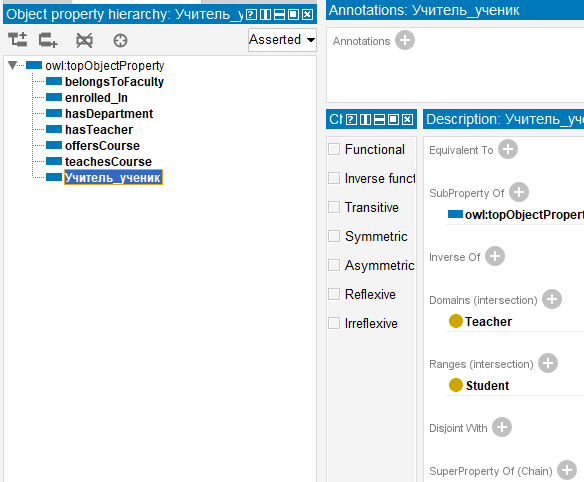
**Создание классов:**



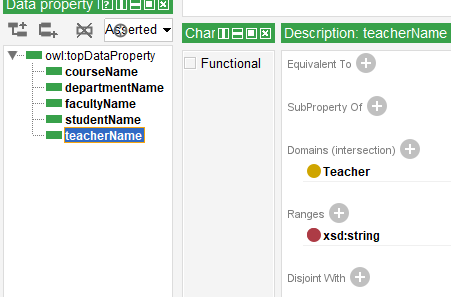
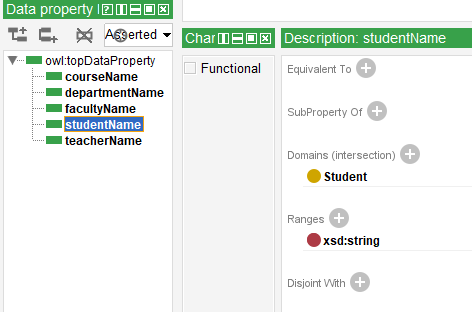
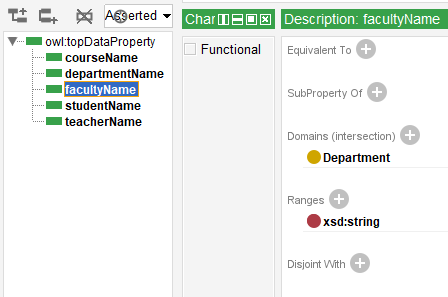
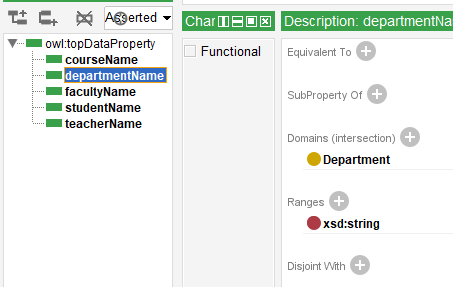
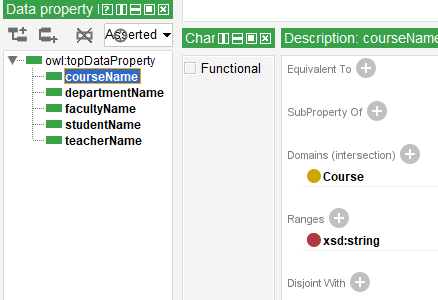
**Создание связей:**



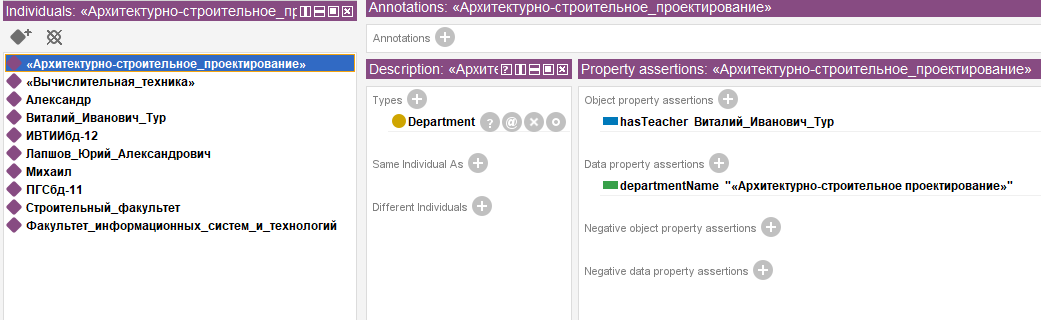


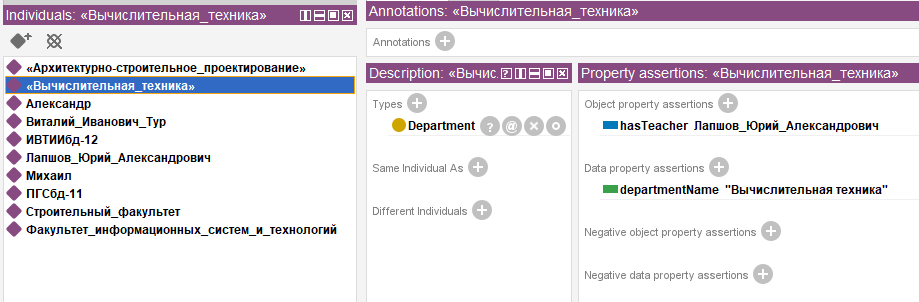


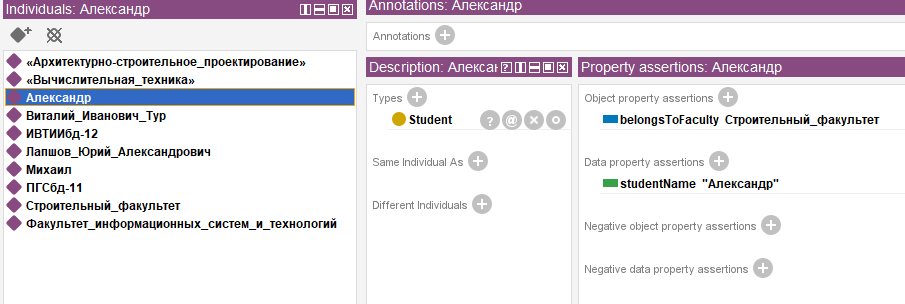
**Создание свойств:**

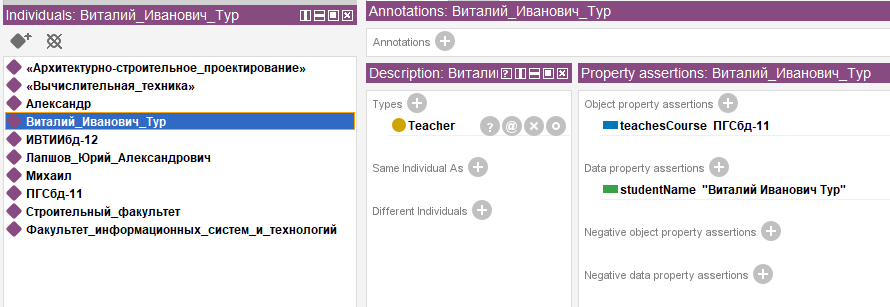


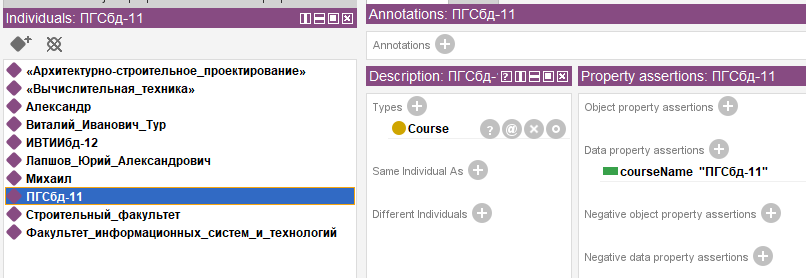
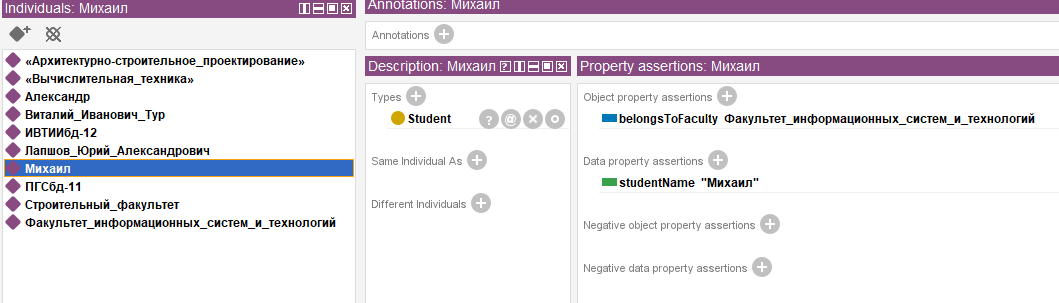
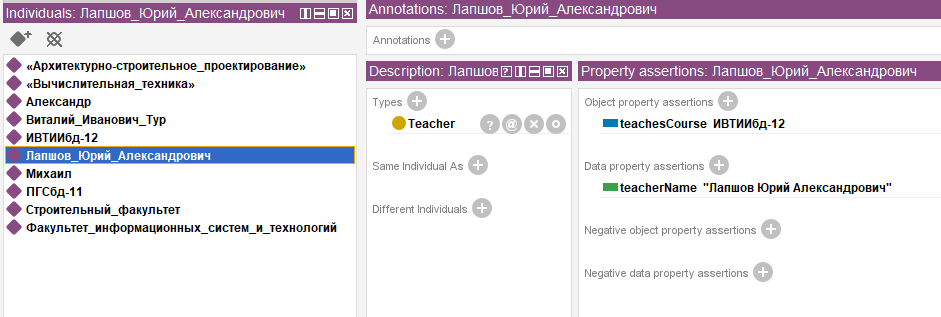
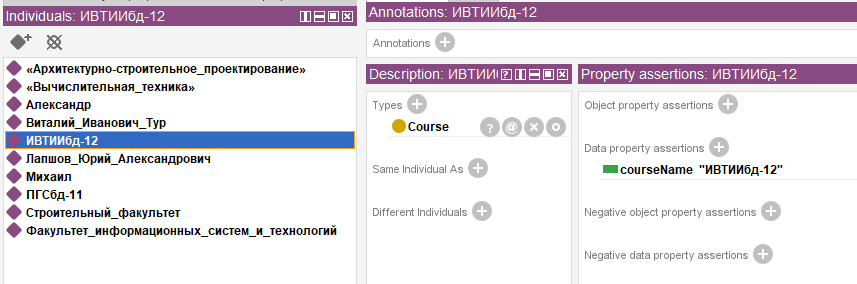
**Создание экземпляров:**

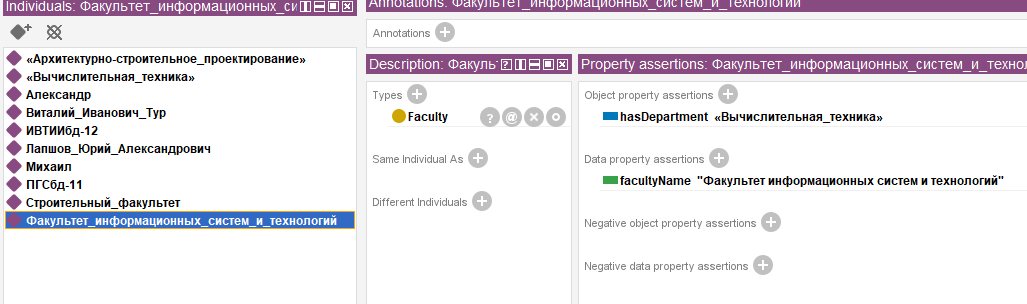
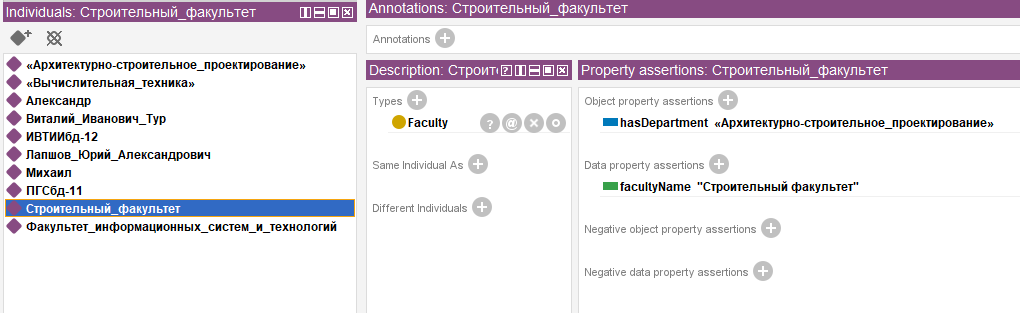




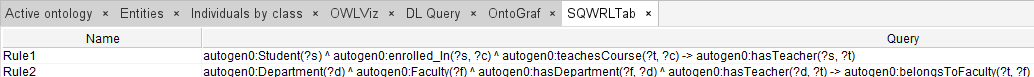




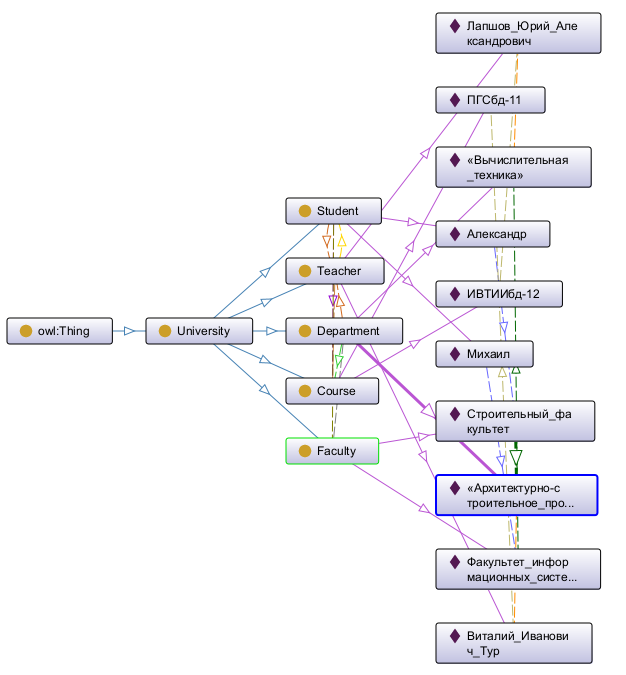




**Правила логического вывода:**

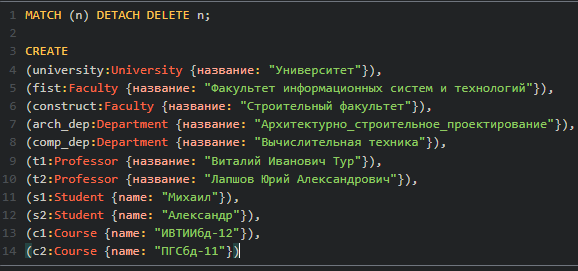


Результат:



**Neo4j**

**Создаем классы:**

MATCH (n) DETACH DELETE n;

CREATE

(university:University {название: "Университет"}),

(fist:Faculty {название: "Факультет информационных систем и технологий"}),

(construct:Faculty {название: "Строительный факультет"}),

(arch\_dep:Department {название: "Архитектурно\_строительное\_проектирование"}),

(comp\_dep:Department {название: "Вычислительная техника"}),

(t1:Professor {название: "Виталий Иванович Тур"}),

(t2:Professor {название: "Лапшов Юрий Александрович"}),

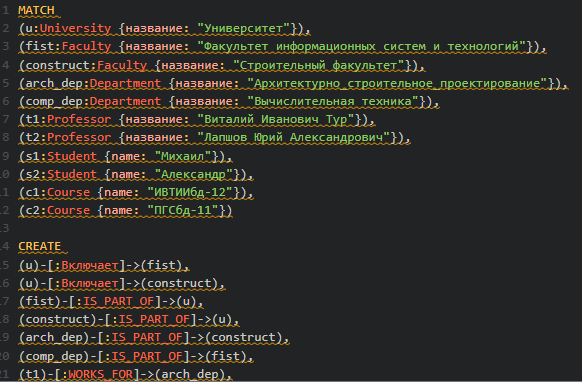
(s1:Student {name: "Михаил"}),

(s2:Student {name: "Александр"}),

(c1:Course {name: "ИВТИИбд-12"}),

(c2:Course {name: "ПГСбд-11"})

**Создаем связи:**



MATCH

(u:University {название: "Университет"}),

(fist:Faculty {название: "Факультет информационных систем и технологий"}),

(construct:Faculty {название: "Строительный факультет"}),

(arch\_dep:Department {название: "Архитектурно\_строительное\_проектирование"}),

(comp\_dep:Department {название: "Вычислительная техника"}),

(t1:Professor {название: "Виталий Иванович Тур"}),

(t2:Professor {название: "Лапшов Юрий Александрович"}),

(s1:Student {name: "Михаил"}),

(s2:Student {name: "Александр"}),

(c1:Course {name: "ИВТИИбд-12"}),

(c2:Course {name: "ПГСбд-11"})

CREATE

(u)-[:Включает]->(fist),

(u)-[:Включает]->(construct),

(fist)-[:IS\_PART\_OF]->(u),

(construct)-[:IS\_PART\_OF]->(u),

(arch\_dep)-[:IS\_PART\_OF]->(construct),

(comp\_dep)-[:IS\_PART\_OF]->(fist),

(t1)-[:WORKS\_FOR]->(arch\_dep),

(t2)-[:WORKS\_FOR]->(comp\_dep),

(s1)-[:STUDIES\_AT]->(fist),

(s2)-[:STUDIES\_AT]->(construct),

(c1)-[:BELONGS\_TO]->(fist),

(c2)-[:BELONGS\_TO]->(construct),

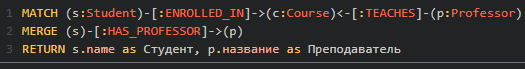
(t1)-[:TEACHES]->(c2),

(t2)-[:TEACHES]->(c1),

(s1)-[:ENROLLED\_IN]->(c1),

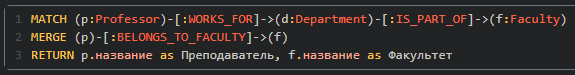
(s2)-[:ENROLLED\_IN]->(c2)

**Создаём правила:**

MATCH (s:Student)-[:ENROLLED\_IN]->(c:Course)<-[:TEACHES]-(p:Professor)

MERGE (s)-[:HAS\_PROFESSOR]->(p)

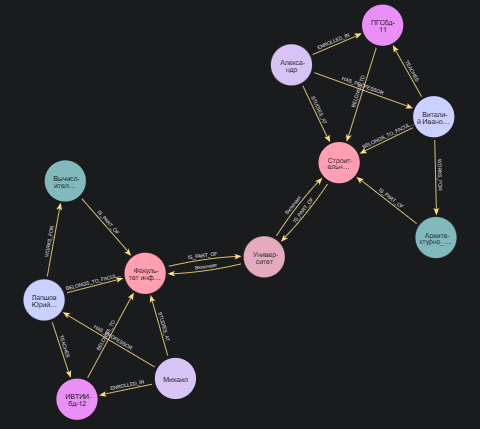
RETURN s.name as Студент, p.название as Преподаватель

MATCH (p:Professor)-[:WORKS\_FOR]->(d:Department)-[:IS\_PART\_OF]->(f:Faculty)

MERGE (p)-[:BELONGS\_TO\_FACULTY]->(f)

RETURN p.название as Преподаватель, f.название as Факультет

**Результат:**



**Вывод**

### ****1.Protege****

**Назначение:**  
Protege — это инструмент для **создания и редактирования онтологий** на основе стандартов **OWL / RDF(S)**.  
Он используется в первую очередь для **семантического моделирования знаний**, а не для хранения или анализа данных.

**Когда использовать Protege:**

* Когда нужно **описать знания** в виде **онтологии** (понятий, классов, свойств, отношений).
* Когда важно **логическое выводение** (reasoning): например, автоматическое определение принадлежности объекта к классу.
* Для **проектирования структуры предметной области** — до этапа внедрения базы данных.
* Подходит для **исследовательских и академических проектов**, где нужно формализовать понятия.

### ****2. Neo4j****

**Назначение:**  
Neo4j — это **графовая база данных**, ориентированная на **работу с конкретными экземплярами данных** и **гибкое хранение связей** между ними.

**Когда использовать Neo4j:**

* Когда нужно **хранить реальные данные** (например, студентов, преподавателей, курсы) и **анализировать связи** между ними.
* Для **поиска путей и зависимостей**: кто обучается у кого, какие факультеты связаны с какими университетами и т.д.
* Для **визуализации и аналитики** сложных сетевых структур.
* В проектах, где важна **производительность запросов по связям** — например, в рекомендательных системах, социальных графах, сетях знаний.

**Заключение:**

**Protege лучше использовать на этапе проектирования модели знаний — чтобы определить, какие сущности существуют, как они связаны и какие логические правила действуют.**

**Neo4j** лучше применять **на этапе реализации и анализа данных**, когда модель уже определена и нужно работать с конкретными объектами и их связями.