



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ

ИУ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА

ИУ7 «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

## КУРСОВАЯ РАБОТА

*НА ТЕМУ:*

*Разработка базы данных для анализа и поиска сообществ в графовой модели социальной сети*

Студент

**ИУ7-64Б**

(группа)

(подпись, дата)

**Парфенов**

(И.О. Фамилия)

Руководитель курсового  
проекта

(подпись, дата)

**Гаврилова Ю.М.**

(И.О. Фамилия)

Консультант

(подпись, дата)

(И.О. Фамилия)

**2025 г.**

# **РЕФЕРАТ**

Расчётно – пояснительная записка 27 с., 5 рис., 10 табл., 6 источн., 1 прил.

## **РАЗРАБОТКА БАЗЫ ДАННЫХ ДЛЯ АНАЛИЗА И ПОИСКА СООБЩЕСТВ В ГРАФОВОЙ МОДЕЛИ СОЦИАЛЬНОЙ СЕТИ**

Цель работы – разработка базы данных для анализа сообществ на графовой модели социальной сети, включая исследование работы алгоритма на разных входных данных.

Был проведён анализ предметной области, была разработана структура базы данных. Было разработано программное обеспечение для анализа сообществ на модели социальной сети. Был разработан алгоритм разметки людей по сообществам. Было проведено исследование работы алгоритма при графах разной разреженности.

# СОДЕРЖАНИЕ

<b>РЕФЕРАТ</b>	<b>4</b>
<b>ВВЕДЕНИЕ</b>	<b>7</b>
<b>1 Аналитическая часть</b>	<b>8</b>
1.1 Анализ предметной области	8
1.1.1 Социальный граф	8
1.1.2 Сообщества	8
1.1.3 Модулярность	8
1.2 Методы поиска сообществ на графах	9
1.2.1 Label Propagation	9
1.2.2 Infomap	9
1.2.3 Edge Betweenness	9
1.2.4 Louvain	10
1.2.5 Сравнение алгоритмов	10
1.2.6 Вывод	10
1.2.7 Формализация данных	11
1.3 Анализ типов баз данных	13
1.3.1 Дореляционные базы данных	13
1.3.2 Реляционные базы данных	13
1.3.3 Постреляционные базы данных	14
1.3.4 Графовые базы данных	14
1.3.5 Сравнительная таблица	15
1.3.6 Вывод	15
<b>2 Конструкторская часть</b>	<b>16</b>
2.1 Требования к программному обеспечению	16
2.2 Проектирование базы данных	16
2.3 Разработка ограничений для обеспечения целостности данных	16
2.4 Разработка ролевой модели	17
2.5 Алгоритм разметки людей по сообществам	17

<b>3</b>	<b>Технологическая часть . . . . .</b>	<b>20</b>
3.1	Средства реализации программы . . . . .	20
3.1.1	Выбор СУБД . . . . .	20
3.1.2	Выбор языка программирования . . . . .	20
3.2	Структура ПО . . . . .	21
3.3	Тестирование . . . . .	21
3.4	Визуализация данных . . . . .	22
<b>4</b>	<b>Исследовательская часть . . . . .</b>	<b>24</b>
4.1	Технические характеристики . . . . .	24
4.2	Исследование . . . . .	24
	<b>ЗАКЛЮЧЕНИЕ . . . . .</b>	<b>26</b>
	<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .</b>	<b>27</b>
	<b>ПРИЛОЖЕНИЕ А . . . . .</b>	<b>28</b>

# **ВВЕДЕНИЕ**

Целью данной работы является разработка базы данных для анализа сообществ на графовой модели социальной сети. Для достижения данной цели необходимо выполнить следующие задачи:

- 1) проанализировать предметную область;
- 2) проанализировать алгоритмы поиска сообществ на графах;
- 3) выбрать наиболее подходящую СУБД для решения поставленной задачи;
- 4) спроектировать базу данных, описать основные её сущности;
- 5) выбрать наиболее подходящие инструменты для разработки ПО;
- 6) исследовать работу алгоритма, на разных входных данных.

# 1 Аналитическая часть

## 1.1 Анализ предметной области

### 1.1.1 Социальный граф

Строгое определение социального графа отсутствует, однако такие графы, как правило, обладают следующими свойствами:

- 1) средняя длина пути между двумя произвольными вершинами невелика; типичным примером являются графы типа «малого мира»;
- 2) для многих вершин выполняется следующее: если вершина А соединена с вершинами В и С, то с высокой вероятностью В и С также соединены между собой;
- 3) граф содержит явно выраженные сообщества.

### 1.1.2 Сообщества

Сообщество в социальном графе представляет собой подмножество вершин, между которыми плотность связей выше, чем с остальными вершинами графа. Примером такого сообщества может служить группа друзей в социальной сети.

### 1.1.3 Модулярность

Модулярность — это численная метрика, характеризующая качество разбиения графа на сообщества. Она определяется по формуле (1.1):

$$Q = \frac{1}{2m} \sum_{i,j} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \delta(C_i, C_j), \quad (1.1)$$

где  $A_{ij}$  — элемент матрицы смежности,  $d_i$  и  $d_j$  — степени вершин  $i$  и  $j$ ,  $m$  — общее количество рёбер в графе,  $\delta(C_i, C_j)$  — индикаторная функция, равная единице, если вершины  $i$  и  $j$  принадлежат одному сообществу, и нулю в противном случае.

## **1.2 Методы поиска сообществ на графах**

### **1.2.1 Label Propagation**

Алгоритм Label Propagation основан на предположении, что вершина, большинство соседей которой принадлежат одному сообществу, с высокой вероятностью принадлежит этому же сообществу. В процессе работы каждая вершина присваивается тому сообществу, которому принадлежит наибольшее количество её соседей. В случае равного числа голосов выбор осуществляется случайным образом.

### **1.2.2 Infomap**

Алгоритм Infomap [1] базируется на концепции случайных блужданий по графу и кодировании траекторий с помощью кодов Хаффмана. Идея заключается в следующем: если использовать иерархическое кодирование, при котором сначала кодируется сообщество, а затем вершина внутри него, можно существенно сократить общую длину кода.

При входе в сообщество записывается его уникальный код, далее — код вершины. При перемещении внутри одного сообщества записываются только коды вершин. При выходе из сообщества фиксируется код выхода. Таким образом, алгоритм стремится минимизировать длину описания случайного блуждания.

### **1.2.3 Edge Betweenness**

Edge Betweenness [1] — это алгоритм, основанный на удалении рёбер с наибольшей центральностью. Для каждой пары вершин определяется множество кратчайших путей. Каждый такой путь делит единичный вес между всеми своими рёбрами. После обработки всех пар вершин каждому ребру сопоставляется значение, равное суммарному количеству проходящих через него кратчайших путей.

Затем из графа последовательно удаляются рёбра с наивысшими значениями edge betweenness. После удаления рёбер образуются компоненты связности, которые интерпретируются как сообщества. Итеративное применение процедуры позволяет построить иерархическую структуру разбиения графа.

### 1.2.4 Louvain

Алгоритм Louvain [2] основан на жадной оптимизации модулярности (1.1). На начальном этапе каждая вершина рассматривается как отдельное сообщество. Алгоритм включает два этапа:

#### **Первый этап:**

- 1) для каждой вершины перебираются все соседние сообщества;
- 2) вершина переносится в сообщество, при котором достигается максимальный прирост модулярности.

#### **Второй этап:**

- 1) строится мета-граф, в котором каждое сообщество представляется одной вершиной. Вес ребра между двумя такими вершинами равен суммарному весу рёбер между соответствующими сообществами. Рёбра внутри одного сообщества трансформируются в петлю;
- 2) алгоритм повторяется с первого этапа.

### 1.2.5 Сравнение алгоритмов

В таблице 1.1 приведена сравнительная характеристика для алгоритмов поиска сообществ на графах.

Таблица 1.1 — Сравнение алгоритмов обнаружения сообществ

Алгоритм	Работает с взвешенными графами	Работает с ориентированными графами	Не требует числа сообществ	Выявляет иерархические сообщества
Label Propagation	+	-	+	-
Infomap	+	+	+	+
Edge Betweenness	+	-	-	-
Louvain	+	-	+	+

### 1.2.6 Вывод

Для описанной задачи в рамках курсовой был выбран алгоритм Louvain, так как он сочетает в себе высокую производительность, способность работать



с взвешенными графами и автоматически выявляет иерархическую структуру сообществ. Алгоритм не требует предварительного указания количества кластеров и хорошо масштабируется на больших графах, что делает его подходящим для моделирования и анализа социальных сетей. Несмотря на то, что он не всегда гарантирует чёткие границы сообществ, его жадная оптимизация модулярности обеспечивает качественные результаты на практике.

### 1.2.7 Формализация данных

Для формального описания структуры используемых данных в рамках курсовой работы была выбрана графовая модель. В данной модели пользователи и сообщества представляются в виде вершин, а связи между ними — в виде рёбер.

Система формализуется как неориентированный взвешенный граф  $G = (V, E)$ , где:

- $V$  — множество вершин. Вершины могут представлять как отдельных пользователей (Person), так и сообщества (Community);
- $E$  — множество рёбер. Рёбра могут быть двух типов: связи между пользователями (FRIENDS\_WITH) и связи принадлежности пользователя к сообществу (BELONGS\_TO);
- каждому ребру  $e \in E$  может быть сопоставлен вес  $w(e)$ , отражающий силу связи. В текущей реализации вес по умолчанию равен 1.

Введены следующие сущности:

- **Person (Человек)** — вершина, соответствующая пользователю социальной сети.
  - id: уникальный идентификатор;
  - name: имя пользователя;
  - level: уровень иерархии (по умолчанию 0).
- **Community (Сообщество)** — вершина, объединяющая пользователей на определённом уровне иерархии.
  - id: уникальный идентификатор сообщества;
  - level: уровень иерархии;
  - members: список идентификаторов участников.
- **FRIENDS\_WITH** — двунаправленная связь между двумя пользовате-

лями, отражающая их взаимодействие.

– weight: вес связи (по умолчанию 1).

— **BELONGS\_TO** — направленная связь между пользователем и сообществом, указывающая на принадлежность.

На рисунке 1.1 изображена диаграмма сущность – связь.

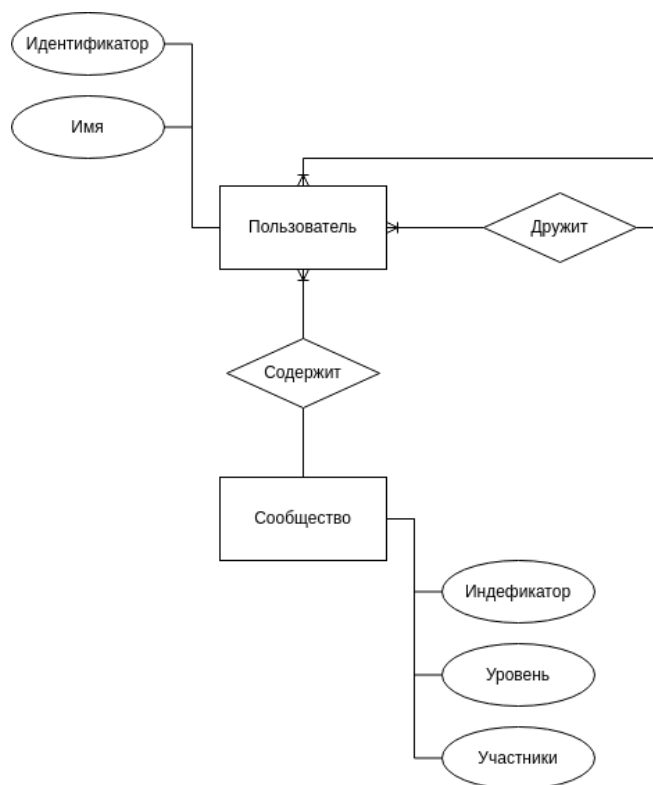


Рисунок 1.1 — Диаграмма сущность – связь

На рисунке 1.2 изображена диаграмма прецедентов.

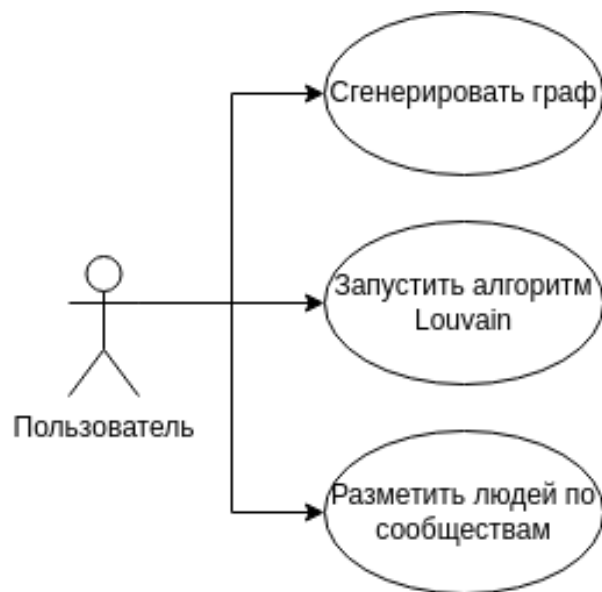


Рисунок 1.2 — Диаграмма прецедентов

### 1.3 Анализ типов баз данных

#### 1.3.1 Дореляционные базы данных

Дореляционные базы данных представляют собой системы управления, разработанные до появления реляционной модели. Наиболее распространённые типы:

- **Иерархические СУБД** — организуют данные в древовидную структуру, где каждая запись имеет строго одного родителя (например, IBM IMS).
  - **Сетевые СУБД** — обеспечивают более гибкие связи между записями в форме графа, что позволяет одной записи иметь несколько родителей.
- Основные недостатки дореляционных моделей:
- жёсткая и негибкая структура хранения данных;
  - высокая степень связанности данных и прикладной логики;
  - затруднённое масштабирование и модификация схем.

#### 1.3.2 Реляционные базы данных

Реляционные СУБД основаны на модели, предложенной Э. Ф. Коддом в 1970 году. Данные представляются в виде таблиц (отношений), где строки

соответствуют записям, а столбцы — атрибутам.

Преимущества реляционного подхода:

- использование декларативного языка запросов SQL;
- логическая независимость данных от физической реализации;
- поддержка нормализации и формального анализа схем данных.

### **1.3.3 Постреляционные базы данных**

Постреляционные базы данных развивают реляционную модель, объединяя её с объектно-ориентированными и семантическими концепциями. Такие СУБД сохраняют поддержку SQL, но дополнительно предоставляют:

- хранение сложных структур данных (массивы, JSON, пользовательские типы);
- объектную иерархию и возможности интеграции с языками программирования;
- расширенные механизмы индексирования и обработки нетабличных данных.

### **1.3.4 Графовые базы данных**

Графовые базы данных предназначены для хранения, обработки и анализа данных, представленных в виде графов. В отличие от реляционной модели, где основными единицами являются таблицы, в графовой модели основными сущностями выступают вершины (объекты) и рёбра (связи между объектами), при этом и те и другие могут содержать произвольные свойства (атрибуты).

Графовые СУБД особенно актуальны в задачах, где важна структура связей между сущностями, например:

- социальные сети;
- рекомендательные системы;
- анализ связности и кластеризация;
- моделирование транспортных и телекоммуникационных сетей.

Преимущества графовых СУБД:

- естественное представление сильно связанных данных;
- высокая производительность при выполнении многосвязных запросов (например, обходов в глубину или по шаблону);

### 1.3.5 Сравнительная таблица

В таблице 1.2 приведена сравнительная характеристика для видов баз данных.

Таблица 1.2 — Сравнение моделей баз данных

Критерий	Дореля- ционная	Реля- ционная	Постреля- ционная	Графовая
Структура хранения	Иерархии или сети	Таблицы (отношения)	Таблицы + расширения	Граф (вершины и рёбра)
Тип связей	Жёстко заданные (один к одному/многим)	Через внешние ключи	Сложные структуры (JSON)	Явные рёбра между вершинами
Язык запросов	Собственные API	SQL	Расширенный SQL, объектные языки	Cypher, Gremlin и др.
Используется	Фиксированные структуры данных	Высокая согласованность данных	Разнородные данные	Сильно связанные данные

### 1.3.6 Вывод

Для решения описанной ранее задачи подходит графовая СУБД. Так как социальная сеть является структурой с сильно связными данными. Так же графовая СУБД позволяет напрямую работать с графовой моделью данных, без создания отдельных таблиц. Язык запросов так же предназначен для работы с графовыми моделями данных, что является плюсом на фоне остальных СУБД.

### Вывод

В данном разделе была проанализирована предметная область, выбран алгоритм для поиска сообществ в графе и подходящий тип СУБД, для решения поставленной задачи.

## 2 Конструкторская часть

По построенной ER диаграмме, можно выделить следующие сущности БД:

### 2.1 Требования к программному обеспечению

Разрабатываемое программное обеспечение должно предоставлять следующие функциональные возможности:

- генерация случайного графа заданной плотности;
- сохранение графа в базу данных;
- выполнение алгоритма Louvain;
- визуализация сообществ.

### 2.2 Проектирование базы данных

В таблице 2.1 представлено описание объекта пользователя.

Таблица 2.1 — Сущность «Пользователь»

Поле	Тип	Описание
id	Целочисленный тип	Уникальный идентификатор пользователя социальной сети
name	Текстовый тип	Имя пользователя

В таблице 2.2 представлено описание объекта сообщества.

Таблица 2.2 — Сущность «Сообщество»

Поле	Тип	Описание
id	Целочисленный тип	Уникальный идентификатор сообщества
level	Целочисленный тип	Уровень сообщества
members	Целочисленный массив	Id членов сообщества

### 2.3 Разработка ограничений для обеспечения целостности данных

Для сущности «Пользователь» было разработано ограничение, которое гарантирует уникальность id для каждой сущности на отдельно взятом уровне.

Для сущности «Сообщество» было разработано ограничение, которое проверяет, что сообщество имеет хотя бы одного участника.

Для связи «Дружба» было разработано ограничение которое гарантирует, что вес связи положителен.

## 2.4 Разработка ролевой модели

Для обеспечения безопасности данных, была разработана следующая ролевая модель:

- **Пользователь** – имеет доступ к функционалу приложения, может запускать алгоритм поиска сообществ, анализировать результаты его работы;
- **Администратор** – имеет все права пользователя, но так же имеет доступ к базе данных, может напрямую писать запросы к бд.

## 2.5 Алгоритм разметки людей по сообществам

После получения супер графа, необходимо каждому человеку сопоставить id тех сообществ в которых он состоит (порядок должен отображать иерархию). Для осуществления этой цели был разработан алгоритм на основе алгоритма «dfs» (*Depth-first search*). Схема алгоритма изображена на рисунке 2.1.

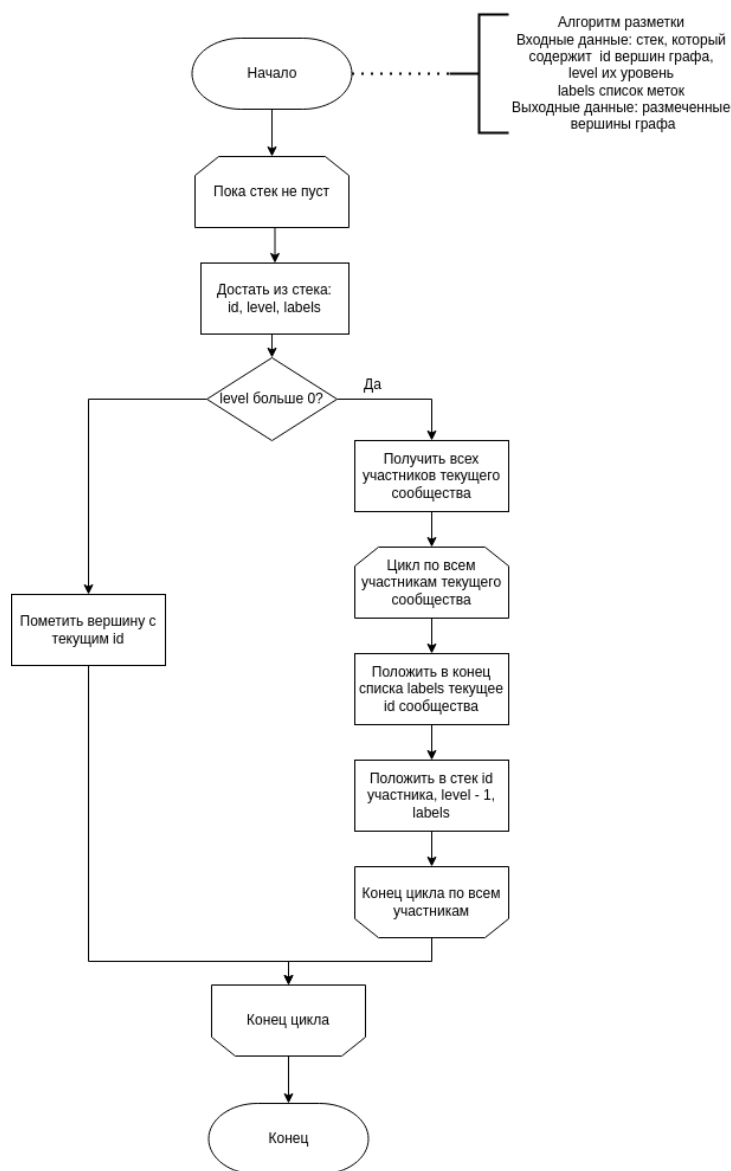


Рисунок 2.1 — Схема алгоритма разметки пользователей социальной сети по сообществам

## Вывод

В рамках конструкторской части были выделены основные сущности и отношения предметной области, отражающие структуру социальной сети и её сообществ. На основании этого была спроектирована база данных, ориентированная на хранение и обработку графовых данных.

Также был разработан алгоритм разметки пользователей по сообществам, учитывающий иерархическую структуру. Использование обхода в глубину (DFS) с явным стеком обеспечивает корректную работу с глубокими уровнями вложенности и позволяет эффективно сопоставлять каждому пользователю все уровни его принадлежности. Предложенная структура и методы обеспечивают



надёжную основу для дальнейшей обработки и анализа социальной графовой модели.

### 3 Технологическая часть

#### 3.1 Средства реализации программы

##### 3.1.1 Выбор СУБД

В таблице 3.1, приведена сравнительная характеристика для различных СУБД, для работы с графовой моделью данных.

Таблица 3.1 — Сравнение графовых СУБД

СУБД	Тип	Язык	ACID	Масштабируемость
Neo4j	Property Graph	Cypher	Да	Вертикальная, кластерная
OrientDB	Multi-model	SQL+Gremlin	Да	Вертикальная, кластерная
ArangoDB	Multi-model	AQL	Да	Горизонтальная
JanusGraph	Property Graph	Gremlin	Частично	Горизонтальная
TigerGraph	Property Graph	GSQL	Да	Горизонтальная

В качестве СУБД была выбрана Neo4j [3], по следующим причинам:

- возможность работы с графовой моделью данных напрямую, без создания дополнительных структур;
- активное сообщество, актуальная документация;
- декларативный язык Cypher, разработанный для нативной работы с графами.

Таким образом, выбор Neo4j обусловлен структурой данных задачи, необходимостью обработки иерархий и сообществ, а также производственными и удобочитаемыми преимуществами графовой модели по сравнению с реляционными СУБД.

##### 3.1.2 Выбор языка программирования

В качестве языка программирования для реализации курсовой работы был выбран язык C# по следующим причинам:

- в стандартной библиотеке языка присутствует поддержка всех структур данных, выбранных по результатам проектирования;
- высокая производительность;

- наличие официального драйвера для взаимодействия с Neo4j;
- наличие актуальной документации.

## 3.2 Структура ПО

На рисунке 3.1 приведена структура разработанного ПО.

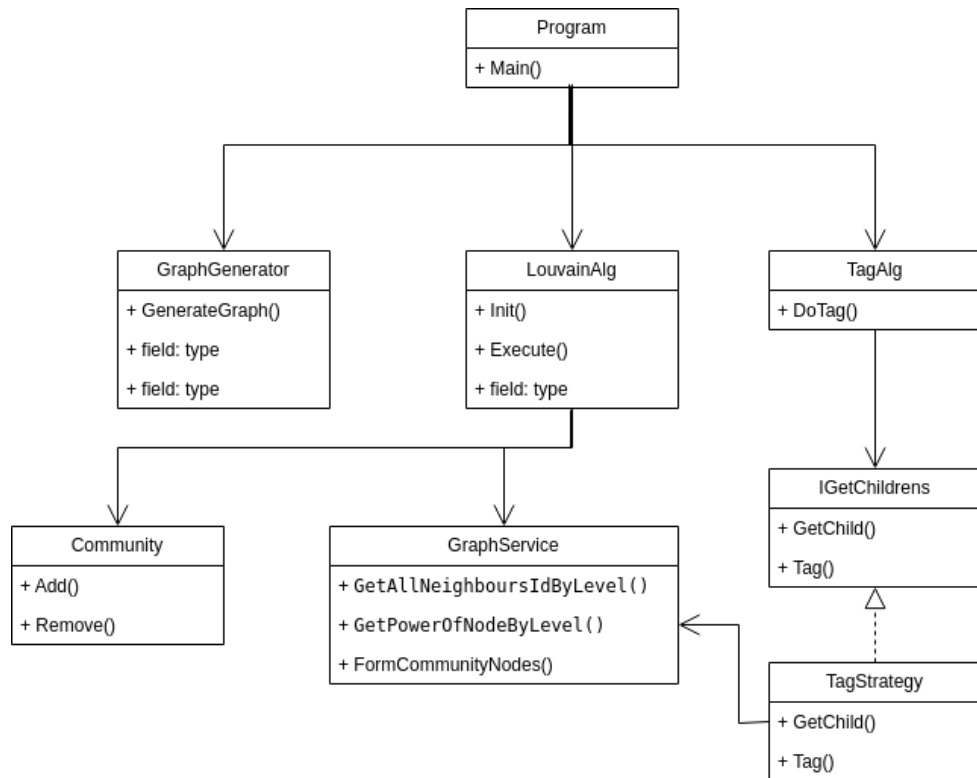


Рисунок 3.1 — Структура ПО

## 3.3 Тестирование

В таблице 3.2 представлены функциональные тесты для реализации алгоритма Louvain.

Таблица 3.2 — Результаты тестирования реализации алгоритма

Номер теста	Параметры (Inter, Intra, количество сообществ)	Результат
1	0.005, 0.2, 10	11
2	0.01, 0.2, 10	9
3	0.05, 0.2, 10	8
4	0.1, 0.2, 3	2
5	0.15, 0.2, 1	1
6	0.005, 0.3, 8	8
7	0.01, 0.3, 10	7
8	0.05, 0.3, 5	5
9	0.1, 0.3, 3	2
10	0.15, 0.3, 1	1
11	0.005, 0.4, 5	5
12	0.01, 0.4, 5	4
13	0.05, 0.4, 5	3
14	0.1, 0.4, 3	1
15	0.15, 0.4, 3	1
16	0.005, 0.5, 3	3
17	0.01, 0.5, 2	2
18	0.05, 0.5, 2	2
19	0.1, 0.5, 1	1
20	0.15, 0.5, 1	1
21	0.005, 0.6, 2	2
22	0.01, 0.6, 5	5
23	0.05, 0.6, 1	1
24	0.1, 0.6, 1	1
25	0.15, 0.6, 1	1

### 3.4 Визуализация данных

Для демонстрации данных был выбран стандартный браузер Neo4j. На рисунке 3.2 приведён пример визуализации данных.

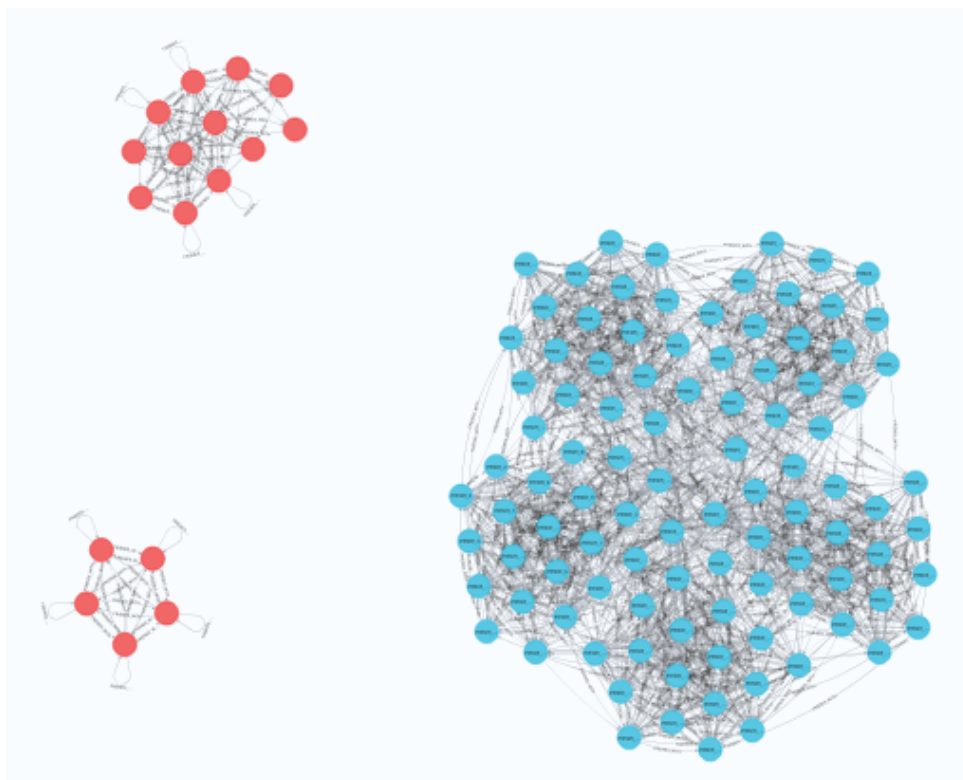


Рисунок 3.2 — Пример визуализации данных

## Вывод

В технологической части был обоснован выбор средств реализации программного обеспечения. В качестве базы данных использована графовая СУБД Neo4j, которая наиболее полно отражает природу предметной области, связанную с моделированием социальных графов и иерархических структур сообществ. Её язык запросов Cypher и встроенная поддержка работы с графами позволили упростить реализацию операций над данными.

Для разработки программной логики выбран язык C# [4], обеспечивающий высокую производительность, удобную работу с асинхронностью и наличие официальной поддержки драйвера для Neo4j.

Визуализация структуры графа и результатов работы алгоритма осуществляется средствами встроенного браузера Neo4j, что позволило наглядно продемонстрировать полученные данные без необходимости использования сторонних решений.

Выбор указанных технологий обеспечил удобство, надёжность и расширяемость при реализации поставленной задачи.

## 4 Исследовательская часть

### 4.1 Технические характеристики

- процессор – Intel® Core™ i5-4590 × 4 [5];
- объём оперативной памяти – 24 ГБ;
- операционная система – Ubuntu 24.04.1 LTS [6].

### 4.2 Исследование

Было проведено исследование зависимости результата работы алгоритма от разреженности графа поданного на вход. В таблицах 4.1 – 4.4 представлены результаты для различных слоёв иерархии сообществ.

- **Intra** (внутрисообщностная плотность) — вероятность наличия ребра между двумя вершинами внутри одного сообщества. Чем выше значение Intra, тем более плотными являются сообщества, то есть больше связей внутри них.
- **Inter** (межсообщностная плотность) — вероятность наличия ребра между вершинами из разных сообществ. Чем выше значение Inter, тем сильнее сообщества связаны между собой, что затрудняет их выявление.

Таблица 4.1 — Количество сообществ на уровне 1

	<b>Inter</b>				
<b>Intra</b>	0.005	0.01	0.05	0.1	0.15
0.2	229	183	103	40	21
0.3	151	135	96	39	18
0.4	84	72	40	25	15
0.5	51	43	30	17	13
0.6	31	27	20	12	8

Таблица 4.4 — Количество сообществ на уровне 4 и выше (усреднённо)

	<b>Inter</b>				
<b>Intra</b>	0.005	0.01	0.05	0.1	0.15
0.2	11	9	8	2	1
0.3	8	7	5	2	1
0.4	5	4	3	1	1
0.5	3	2	2	1	1
0.6	2	2	1	1	1

Таблица 4.2 — Количество сообществ на уровне 2

	<b>Inter</b>				
<b>Intra</b>	0.005	0.01	0.05	0.1	0.15
0.2	30	22	9	5	1
0.3	21	16	8	4	1
0.4	10	8	5	2	1
0.5	6	5	3	2	1
0.6	5	3	2	1	1

Таблица 4.3 — Количество сообществ на уровне 3

	<b>Inter</b>				
<b>Intra</b>	0.005	0.01	0.05	0.1	0.15
0.2	19	14	8	2	1
0.3	11	9	7	2	1
0.4	6	5	4	1	1
0.5	4	3	2	1	1
0.6	2	2	1	1	1

## Вывод

По данным, полученным в ходе исследования, можно сделать следующие выводы:

- при увеличении значения параметра Inter (межсообщностной плотности) количество обнаруженных сообществ на всех уровнях иерархии значительно уменьшается. Это указывает на снижение чёткости границ между сообществами при росте количества межсообщностных связей;
- при увеличении значения Intra (внутрисообщностной плотности) также наблюдается уменьшение количества сообществ, особенно на верхних уровнях иерархии. Это может быть связано с тем, что более плотные внутренние связи способствуют укрупнению сообществ;
- на более глубоких уровнях иерархии количество сообществ резко сокращается, особенно при высоких значениях параметров Inter и Intra. Это свидетельствует о том, что структура сообщества становится менее детализированной.

## ЗАКЛЮЧЕНИЕ

Цель работы была достигнута: была разработана база данных для анализа сообществ на графовой модели социальной сети. Для достижения данной цели были выполнены следующие задачи:

- 1) проанализирована предметная область;
- 2) проанализированы алгоритмы поиска сообществ на графах;
- 3) выбрана наиболее подходящая СУБД для решения поставленной задачи;
- 4) спроектирована база данных, описаны её основные сущности;
- 5) выбраны наиболее подходящие инструменты для разработки ПО;
- 6) исследована работа алгоритма, на разных входных данных.

В результате исследования было выяснено, что число обнаруживаемых сообществ существенно зависит от плотности связей в графе: увеличение как межсообщностной, так и внутрисообщностной плотности приводит к укрупнению сообществ и снижению их количества, особенно на более высоких уровнях иерархии.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методы выделения сообществ в социальных графах. Никишин Евгений Сергеевич. [http://www.machinelearning.ru/wiki/images/8/8a/Nikishin\\_coursework\\_community\\_detection.pdf](http://www.machinelearning.ru/wiki/images/8/8a/Nikishin_coursework_community_detection.pdf)
2. On the Power of Louvain in the Stochastic Block Model [Электронный ресурс]. Режим доступа: <https://groups.csail.mit.edu/tds/papers/Mallmann-Trenn/NEURIPS2020.pdf> (Дата обращения 2.06.25)
3. Документация Neo4j [Электронный ресурс]. Режим доступа: <https://neo4j.com/docs/> (Дата обращения 2.06.25)
4. Документация по C# [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (Дата обращения 2.06.25)
5. Документация к процессору [Электронный ресурс]. Режим доступа: <https://www.intel.com/content/www/us/en/products/sku/80815/intel-core-i54590-processor-6m-cache-up-to-3-70-ghz/specifications.html> (Дата обращения 2.06.25)
6. Сайт дистрибутива [Электронный ресурс]. Режим доступа: <https://ubuntu.com/> (Дата обращения 2.06.25)

## **ПРИЛОЖЕНИЕ А**

Презентация к курсовой работе состоит из 9 слайдов.