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

ФАКУЛЬТЕТ \_\_\_\_\_\_\_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_Системы обработки информации и управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***Разработка синтаксической основы навигационного \_\_\_\_\_языка для работы с архиграфовой СУБД\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_ИУ5-32М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_**А. Р. Миронова**\_

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Е. Гапанюк\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2024 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_ИУ-5\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_ В.И. Терехов

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме Разработка синтаксической основы навигационного языка для работы с \_\_\_\_\_\_\_\_\_\_\_\_архиграфовой СУБД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы ИУ5-32М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Миронова\_Александра\_Романовна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_ учебная \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_учебная тематика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к 12 нед., 50% к 14 нед., 75% к 15 нед., 100% к 16 нед.

***Техническое задание*** Разработка синтаксической основы навигационного языка для работы \_с архиграфовой СУБД***\_***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 28 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « 9 » \_сентября 2024 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_ Ю.Е. Гапанюк

(Подпись, дата И.О.Фамилия)

**Студент**  \_\_\_\_\_\_\_\_\_\_. А.Р. Миронова

(Подпись, дата И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**1. Оглавление**

[2. Введение 4](#_Toc185982986)

[3. Постановка задачи 4](#_Toc185982987)

[4. Актуальность 5](#_Toc185982988)

[5. Описание предметной области 6](#_Toc185982989)

[5.1. Общие сведения о метаграфах 6](#_Toc185982990)

[5.2. Понятия протографа и архиграфа 8](#_Toc185982991)

[5.3. Разработка архиграфой модели 10](#_Toc185982992)

[5.4. Хранение универсальной модели данных 11](#_Toc185982993)

[6. Описание необходимого функционала навигационного языка для работы с архиграфовой СУБД 15](#_Toc185982994)

[7. Анализ существующих графовых языков 16](#_Toc185982995)

[7.1. G 16](#_Toc185982996)

[7.2. GraphLog 18](#_Toc185982997)

[7.3. G-CORE 19](#_Toc185982998)

[8. Описание разрабатываемого навигационного языка 20](#_Toc185982999)

[8.1. Функциональность 20](#_Toc185983000)

[8.2. Синтаксис 22](#_Toc185983001)

[9. Заключение 26](#_Toc185983002)

[10. Список использованной литературы 26](#_Toc185983003)

# 2. Введение

В настоящее время, с увеличением объемов данных и развитием технологий, возникла необходимость разработки универсальных систем управления базами данных и использования новых методов и инструментов для работы с ними. СУБД, рассматриваемая в данной работе, основана на архиграфовом подходе, позволяющем работать с разными структурами данных, такими как таблицы, графы, документы, индексы и многомерные кубы. В работе рассматриваются основные принципы работы архиграфовой СУБД, ее возможности и преимущества по сравнению с другими системами управления.

В современном мире базы данных играют важную роль в хранении и управлении большим объемом информации. Однако разработка и оптимизация систем управления базами данных (СУБД) является сложной задачей, требующей специализированных знаний и навыков. Одним из важных аспектов разработки СУБД является язык программирования, на котором будут писаться запросы и операции с базой данных. В данной работе рассматривается собственный навигационный язык для работы с архиграфовой СУБД, взаимодействующий с обработчиками разных структур данных. Будут рассмотрены основные возможности и преимущества данного языка, а также приведены примеры его синтаксиса.

# 3. Постановка задачи

Цель данной работы состоит в том, чтобы проанализировать существующие системы управления базами данных, выявить их преимущества и недостатки, и на основе результатов анализа спроектировать навигационный язык для собственной СУБД, ориентированной на работу с разными структурами данных на основе различных обработчиков.

В ходе выполнения работы будут выполнены следующие задачи:

* изучить предметную область: метаграфовую и основанную на ней архиграфовую модели, способы хранения универсальной модели данных в метаграфовой СУБД;
* рассмотреть существующие системы управления базами данных и языки для работы с ними;
* определить функции собственного навигационного языка архиграфовой СУБД и описать его синтаксис.

# 4. Актуальность

Разработка систем управления базами данных является актуальной темой в современном информационном обществе. Постоянное увеличение объемов данных и разнообразие их типов и источников представляют новые вызовы для эффективной обработки и управления данными.

Системы управления базами данных являются неотъемлемой частью информационных систем и бизнес-процессов в различных сферах деятельности - от финансовых учреждений до здравоохранения и маркетинга. Оптимальное проектирование и разработка СУБД позволяют обеспечить быстрый доступ к данным, высокую производительность и надежность системы, а также легкость в масштабировании и поддержке.

Кроме того, с развитием новых технологий, таких как облачные вычисления, машинное обучение, интернет вещей (IoT) и большие данные (Big Data), появляются новые требования к функциональности и производительности СУБД. Разработка СУБД, работающей с данными, которые хранятся в наиболее выгодных форматах, позволяет значительно повысить эффективность и производительность СУБД, ускорив обработку данных.

Таким образом, разработка систем управления базами данных остается актуальной задачей, способствующей оптимизации работы с данными, улучшению качества принятия решений и достижению конкурентных преимуществ в современной информационной среде.

В данной работе рассмотрены все аспекты предметной области, связанной с разработкой архиграфовой СУБД, изучены существующие системы управления базами данных, и разработаны функции для собственного навигационного языка, работающего с разрабатываемой СУБД.

# 5. Описание предметной области

## 5.1. Общие сведения о метаграфах

Разрабатываемая архиграфовая СУБД является надстройкой над метаграфовой СУБД, поэтому, в первую очередь, необходимо дать пояснения метаграфам, положившим основу рассматриваемой СУБД.

Метаграфы – тип модели данных, который имеет возможность работать с данными на верхних обобщенных уровнях и обращаться к деталям только тогда, когда в этом возникнет необходимость. Таким образом, на основе метаграфов можно создать универсальную модель данных, используемую для структурирования очень большого объема данных благодаря предоставлению неограниченного количества уровней вложенности данных и инкапсуляции вложенных данных.

Впервые термин «метаграф» был упомянут в монографии А. Базу и Р. Бланнинга [1]. Их определение метаграфа включало:

* Возможность объединять вершины в произвольные группы и внутри этих групп иметь вложенные группы вершин.
* Возможность соединять ребрами как отдельные вершины, так и группы вершин, включая любую вложенность, проникающую через границы групп.
* Наличие переменных на ребрах, которым можно присвоить значения.

В дальнейшем появлялись различные модификации метаграфовой модели: модель с метавершинами [2], иерархическая модель с метавершинами и метарёбрами [3], аннотируемая модель [4, 5]. Расширение модели Базу-Бланнинга, предложенное в аннотируемых метаграфах является наиболее универсальной из представленных моделей и представляет интерес для её использования в рамках универсальной модели данных. Аннотируемой данную модель назвали, потому что метавершины или метаребра содержащие те же внутренние объекты, что и другие метавершины или метаребра, аннотируют эти объекты, позволяя добавить к ним некоторые дополнительные атрибуты в новом представлении.

Опишем основные элементы аннотируемого метаграфа. Сам **метаграф** определяется как: , где – метаграф, – множество вершин метаграфа, MV – множество метавершин метаграфа, E – множество рёбер метаграфа, ME – множество метарёбер метаграфа.

**Вершина** определяется как: , где – вершина метаграфа, – атрибуты вершины.

**Ребро** метаграфа описывается как: , где – ребро метаграфа, – исходная вершина (метавершина) ребра, – конечная вершина (метавершина) ребра, – атрибуты ребра, – признак направленности ребра ( = true – направленное ребро; = false – ненаправленное ребро).

**Метавершина** задаётся как: , где – метавершина метаграфа, – атрибуты метавершины, – фрагмент метаграфа.

**Фрагмент метаграфа** в общем виде задаётся как: , где – фрагмент метаграфа, – элемент, который является либо ребром, либо метаребром, либо вершиной, либо метавершиной.

**Метаребро** задаётся как: , где – метаребро метаграфа, – исходная вершина (метавершина) ребра, – конечная вершина (метавершина) ребра, – атрибуты ребра, – признак направленности ребра ( = true – направленное ребро; = false – ненаправленное ребро), – фрагмент метаграфа.

Таким образом в состав метаграфа аннотируемой модели входят рёбра, вершины, метарёбра и метавершины. Каждый элемент имеет свой набор атрибутов, где каждый атрибут имеет имя и значение. Рёбра и метарёбра такого метаграфа могут проникать через границы метавершин и метарёбер на любую глубину вложенности. Пример аннотированного метаграфа показан на рис. 1.

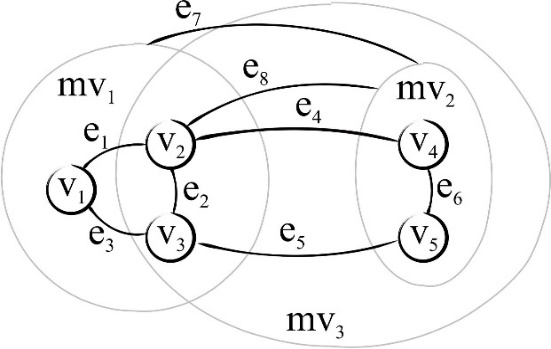


Рис. 1. Пример аннотируемого метаграфа

## 5.2. Понятия протографа и архиграфа

Метаграфовая модель имеет широкие возможности её применения, но в контексте разрабатываемой модели их недостаточно для решения всех проблем. Разрабатываемая СУБД должна иметь возможность обрабатывать реляционные данные, данные баз данных NoSQL, многомерные кубы, или/и текстовые документы для поисковых индексов, поэтому их описание с помощью метаграфовой модели является достаточно трудной задачей. Следовательно, необходимо расширить эту модель. Для этого обратимся к понятиям протографа и архиграфа, предложенных в [6, 7], где рассматривалась концепция, позволяющая описывать различные обобщения графов (метаграфы, гиперграфы, мультиграфы и другие) через архиграф и протограф.

**Архиграфом** называют набор множеств, между элементами которых существует отношение инцидентности. Формально архиграф задаётся как: , где – архиграф, – множество элементов, n – число множеств. Таким образом, можно сказать, что архиграф состоит из некоторого числа классов, где содержит множество элементов i-го класса. Примером архиграфа 2 степени:, – является обычный граф, заданный как: , где – множество рёбер, – множество вершин.

**Протографом** называется множество элементов и матрица их соседства:, – в которой 1 означает наличие соседства элемента с элементом , а 0 его отсутствие. Протограф можно рассматривать как граф, не имеющий ребер; роль ребер выполняет прилегание вершин друг к другу. Примерами протографов являются: стек, очередь, карта. Протограф может быть как ориентированным, так и неориентированным. Пример каждого протографа изображены на рисунках 2.а) и 2.б) соответственно.

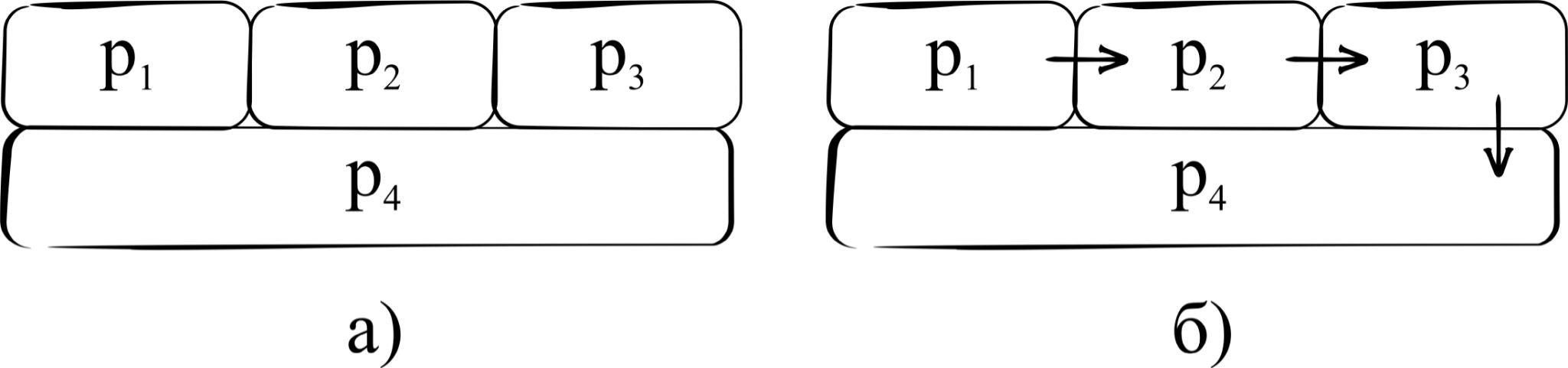


Рисунок 2. Пример протографа

а) неориентированный протограф, б) ориентированный протограф

Протограф является минимальной моделью, и за счёт выделения подмножеств можно сформировать граф, метаграф, архиграф. Архиграф можно определить как протограф, элементы которого разбиты на n классов. Также в работе [6] детально описано представление различных обобщений графа в виде протографа, в том числе и метаграф.

Таким образом, ранее описанные метаграфовые модели можно систематизировать через понятие архиграфа. Так первая модель, предложенная А. Базу и Р. Бланнингом [1], является архиграфом 4 степени и может быть представима в виде протографа из 4 классов: вершины, группы вершин, рёбра и переменные. А аннотируемая модель является архиграфом со степенью 5 и может быть представлена в виде протографа из 5 классов: вершины, метавершины, рёбра, метарёбра и атрибуты.

Следовательно, мы можем расширять архиграфовое представление аннотируемой модели метаграфа до архиграфа более высокой степени.

## 5.3. Разработка архиграфой модели

Для описания универсальной модели данных используется архиграф, основанный на архиграфе аннотируемой метаграфовой модели с добавлением новых классов для описания неестественных для метаграфа форматов.

Далее будем рассматривать универсальную модель данных, поддерживающую графовое, табличное и многомерное представления данных, а также поисковой индекс.

Начнём с табличного представления. В работе [8] предлагалось учесть в архиграфе как таблицу, так и все её элементы. Такой подход усложняет процесс чтения таблицы по сравнению с представлением её как набора последовательных байт из-за того, что выделенные в отдельные элементы архиграфа элементы таблицы потребуют дополнительных ресурсы на их поиск и чтение. Следовательно, будет достаточно добавить в архиграф один класс «таблицы».

Для многомерного представления также выделим один класс «многомерный куб».

Для поискового индекса нам достаточно выделить класс «индекс» для описания самого индекса и класс «документ» для описания документов, связанных с индексом.

Таким образом, для описания универсальной модели данных озера данных с поддержкой графовых, табличных, многомерных данных и поисковых индексов потребуется архиграф с 9 классами: вершины, рёбра, метавершины, метарёбра, многомерные кубы, таблицы, индексы, документы, атрибуты.

Также необходимо задать формализованную систему правил смежности элементов указанных классов в соответствующем архиграфу протографе:

* Каждое ребро может быть смежным с одним из элементов следующих классов: вершины, метавершины, таблицы, многомерные кубы, индексы, документы.
* Каждое метаребро может быть смежным с одним из элементов следующих классов: вершины, метавершины, таблицы, многомерные кубы, индексы, документы.
* Метавершины могут содержать внутри себя: вершины, метавершины, ребра, метаребра, таблицы, многомерные кубы, документы, индексы.
* Метаребра могут содержать внутри себя: вершины, метавершины, ребра, метаребра, таблицы, многомерные кубы, документы, индексы.
* Атрибуты могут быть смежными с элементом одного из классов: вершин, метавершин, рёбер, метарёбер, таблиц, многомерных кубов, индексов, документов.

Предложенная универсальная модель данных на основе архиграфа позволит использовать сложные метаграфовые структуры и привязывать к ним таблицы, многомерные кубы и поисковые индексы. Это даст возможность работать на единой структуре данных всем основным видам приложений: транзакционным системам, использующим сейчас реляционные, графовые или NoSQL базы данных, аналитическим системам, использующим многомерные структуры данных, средствам поиска по образцам текстов, системам управления мастер-данными и приложениям интернета вещей.

## 5.4. Хранение универсальной модели данных

Исследование, проводимое в данной работе, является частью большого проекта, поэтому важно рассмотреть схему данного проекта в целом, а затем определить конкретные решаемые задачи. В рамках проекта необходимо реализовать систему создания и поддержки озер данных на основе универсальной модели данных согласно архитектуре, представленной на рисунке 3.

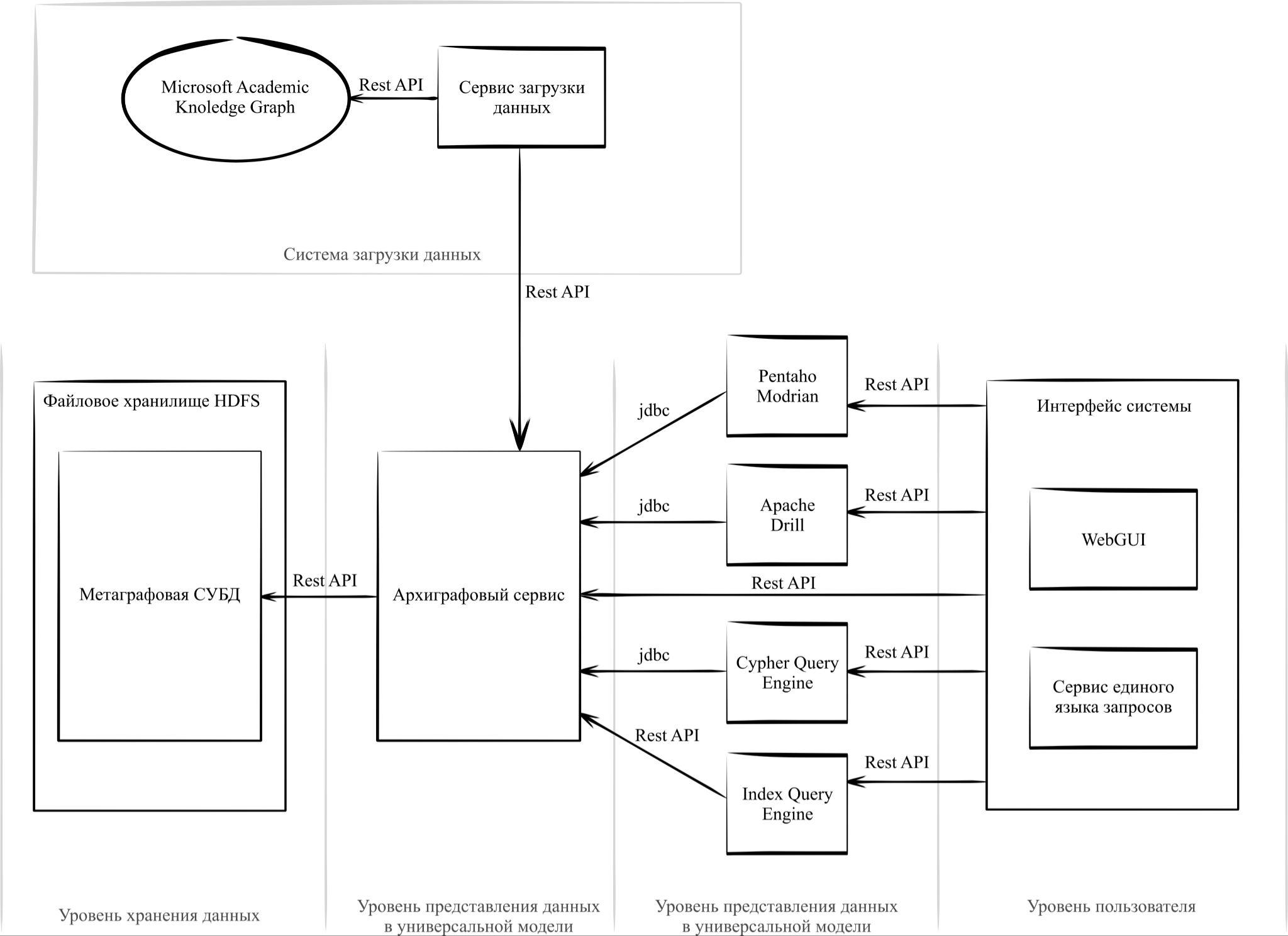


Рисунок 3. Возможная схема озера данных на основе универсальной модели данных

Для хранения архиграфа используется специальная метаграфовая СУБД, реализуемая в рамках отдельного подпроекта. Примеры таких СУБД были предложены в [9, 10, 11].

Ядро архитектуры системы создания и поддержки озер данных будет содержать 3 основных уровня:

* уровень хранения данных (хранение всех данных озера в метаграфовой СУБД, развёрнутый на основе файлового хранилища HDFS),
* уровень представления данных (организация структуры хранения данных в метаграфовой СУБД, построение архиграфа и предоставления интерфейсов для доступа к нему),
* уровень обработки аналитических запросов (интерпретация популярных вариантов обращения к представлениям данных: SQL-запрос, MDX-запрос, запрос на поиск по по образцу текста и Cypher-запрос).

Ядро предложенной системы представляет собой архиграфовую СУБД. Помимо трёх уровней описывающих ядро системы имеет смысл реализовать **уровень пользователя**, отвечающий за отправку запросов данными, либо через универсальный язык запросов к архиграфовой СУБД, либо через пользовательский интерфейс, позволяющий визуализировать результатов запросов, и **систему загрузки данных**, отвечающую за извлечение и загрузку в систему необходимых данных.

При реализации уровня обработки аналитических запросов, может потребоваться выделение дополнительных сервисов-адаптеров для согласования внешнего интерфейса уровня представления данных и форматов запроса данных от обработчиков запросов.

Для работы со всей совокупностью обработчиков запросов предполагается разработать единый навигационный язык для обращения к архиграфовой СУБД с внутренними секциями для определённого обработчика данных. Данный язык подробно рассматривается в данной работе.

В соответствии с описанной архитектурой предлагается хранить данные для архиграфа в метаграфе. Опишем вариант хранения в метаграфе ранее описанных архиграфовых классов, за исключением классов, которые уже являются элементами метаграфа: «метавершины», «вершины», «рёбра», «метарёбра», «атрибуты».

Элементы класса «таблицы» могут быть представлены обычными вершинами метаграфа, с указанием в атрибутах информации о названии таблицы, имени и типе полей, ключах, данные таблицы. Связи между таблицами могут быть реализованы через реляционные отношения с помощью первичных и вторичных ключей или через ребро метаграфа связывающее таблицы между собой. Пример представления таблицы показан на рисунке 4.a).

Элементы класса «документы» могут быть представлены обычными вершинами метаграфа, с указанием в атрибутах информации о названии документа, и данные документа. Пример представления документа показан на рисунке 4.b).

Элементы класса «индексы» также могут быть представлены в виде вершин метаграфа с указанием в атрибутах информации о названии индекса, и данные индекса. Пример представления индекса также показан на рисунке 4.c).

Элементы класса «многомерные кубы» могут быть представлены как метавершина, содержащая в себе множество технических вершин, описывающих оси куба. Тогда технические вершины осей куба будут содержать в атрибутах название оси, данные со значениями осей и данные с описанием иерархии, а метавершина куба будет содержать в атрибутах название куба, данные с мерами и описанием полей в мерах. Пример представления куба с 3 осями показан на рисунке 4.d).

Для того, чтобы отличать элементы классов архиграфа, хранящихся в одинаковых элементах метаграфа, для каждого элемента вводится атрибут «type», указывающий на тип соответствующего элемента. На рисунке 4.a), 4.b), 4.c) и 4.d) у всех элементов указывается данный атрибут.

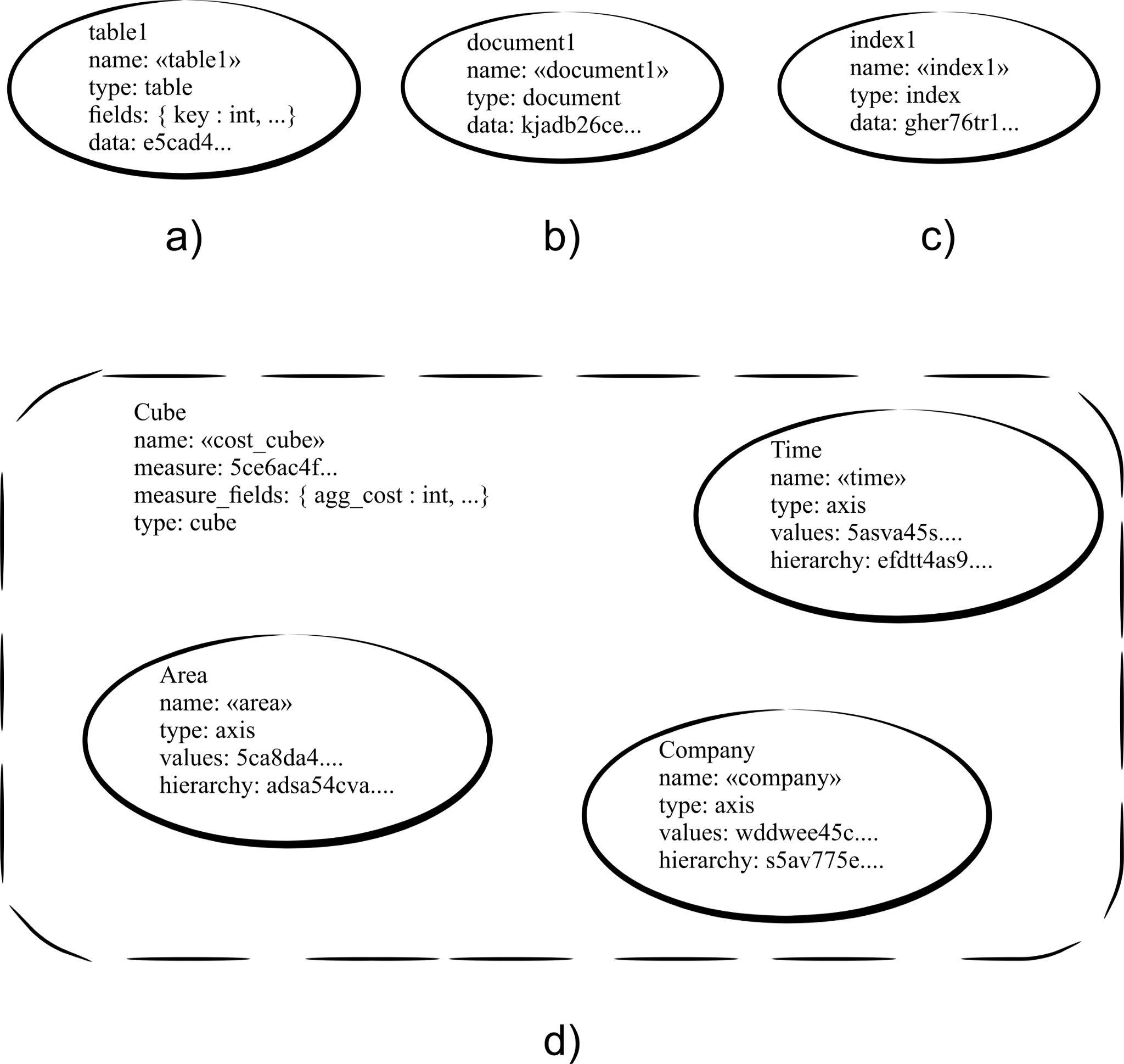


Рисунок 4. Вариант представления в метаграфе классов:

a) таблиц, b) документов, c) индексов, d) многомерных кубов

Таким образом можно использовать метаграфовую СУБД для хранения данных универсальной модели данных. Такой способ также позволяет внутри метаграфа связывать разные представления между собой или с какими-либо метаданными о них.

# 6. Описание необходимого функционала навигационного языка для работы с архиграфовой СУБД

Теперь, имея представление о структуре проекта, определим функционал навигационного языка:

1. Получение запросов от интерфейса пользователя через API.
2. Обработка запроса интерфейса с целью определения обработчика: SQL (для таблиц), MetaCypher (для графов), MDX (для многомерных кубов) и обработчика поисковых индексов (для обработки документов).
3. Навигация по элементам метаграфовой БД модели.
4. Возможность совершать запросы (на получение, изменение, удаление данных) к метаграфовой БД.
5. Возможность перехода по вершинам архиграфовой модели и вхождения в эти вершины (в случае, если это метавершины) для последующей их обработки в зависимости от их типа.
6. Отправка запросов соответствующим обработчикам для их исполнения: запросы данных обработчиков выполняются на серверах Pentaho Mondrian (MDX), Cypher Query Engine (MetaCypher), Apache Drill (SQL) и Index Query Engine (поисковые индексы) соответственно.
7. Получение результата от обработчиков запросов.
8. Сохранение элементов с данными в архиграфовом сервисе в переменных соответствующего типа (вершина, ребро, метавершина, метаребро и другие). Также сюда можно добавить возможность создания массивов из этих переменных.
9. Отправка результата запроса пользовательскому интерфейсу по API.

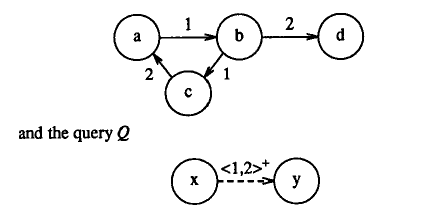
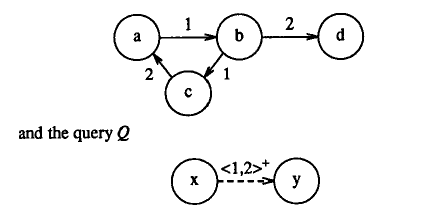
# 7. Анализ существующих графовых языков

Наши исследования привели нас к заключению, что навигационные элементы есть даже в самых сложных и распространенных декларативных языках. Однако кроме как в теоретических разработках навигационные языки в чистом виде никто не использует. Поскольку гораздо выгоднее переложить часть работы на компьютер, добавляя декларативность. В нашем случае, мы делаем Proof-of-Concept проект-исследование, поэтому, мы можем позволить себе ограничиться разработкой и использованием навигационного языка.

## 7.1. G

Самым первым навигационным языком, найденным нами в доступных источниках, является G [12]. Это визуальный язык запросов, основанный на регулярных выражениях, которые позволяют легко формулировать рекурсивные запросы. Запросы в языке G задаются графически. Графический запрос состоит из набора направленных маркированных мультиграфов. Метками их узлов Q могут быть переменные или константы, а метками ребер - регулярные выражения, определенные над n-кортежами переменных и констант. Результатом выполнения запроса является набор мультиграфов, которые соответствуют подграфам исходного графа.

Помимо графического задания запросов авторы языка G так же предлагают записывать эквивалентное строковое представление запроса на языке Prolog. И использовать компилятор программ C-Prolog для выполнения запросов. Приведем пример графического запроса на языке G и выполнения эквивалентного запроса на языке Prolog. На рисунке 5а представлен исходный граф. Граф содержит 4 вершины: a,b,c,d; соединенные направленными ребрами с метками 1 и 2. На рисунке 5б представлен графический запрос к этому графу Он состоит из двух вершин, которые обозначены переменными x и y и положительного замыкания регулярного выражения <1,2>+.



а) б)

Рисунок 5. Запрос на языке G к графу:  
а) исходный граф, б) графическое представление запроса

Далее представлен запрос на языке Prolog, эквивалентный запросу на языке G, представленному на рисунке 2б:

simple1(X, Y, Visited, [X/Visited ]) –

r(X,Y,1),

not member(X, Visited)

simple2(X. Y, Visited, [X/Visited ]) -

r(X,Y,2),

not member(X, Visited)

sequence(X, Y, Visited, NewVisited) -

simple1(X, 2, Visited, NV),

simple2(Z, Y, NV, NewVisited)

closure(X, Y, Visited, New Visited) -

sequence(X, Y, Visited, NewVisited)

closure(X, Y, Visited, NewVisited) -

sequence(X, Z, Visited, NV),

closure (Z. Y, NV, New Visited)

s(X, Y) -

closure(X, Y, []. Visited),

not member(Y, Visited)

Среди недостатков навигационных возможностей языка G отмечено отсутствие поддержки циклов, поиска кратчайшего пути или вычисления расстояния между узлами, поддержки агрегатных функции.

## 7.2. GraphLog

Еще одну разработку навигационного языка представляют авторы работы [13]. Graphlog представляет собой адаптацию языка DataLog для графовых запросов. Запросы на языке Graphlog тоже задаются графически и представляют собой шаблоны гафов. Такой шаблон содержит переменные-вершины, вместо которых будут подставлены реальные вершины исходного графа, подходящие под условия шаблона. В шаблоне так же используются ребра, которые либо представляют базовое отношение, либо сами определены в другом графе запроса. Ребра в графе запроса тоже соответствуют ребрам или путям в графе базы данных. Для определения этих путей используются регулярные выражения. В отличие от языка G в запросах GraphLog можно использовать предикаты. Запрос на языке GraphLog с помощью функции логической трансляции преобразуется в предикатный вид языка DataLog, выполняется на нем как программа и затем обратно возврящается в графический вид GraphLog. Таким образом, результатом выполнения запроса является графическое представление. Рассмотрим пример графического запроса на языке GraphLog (рисунок 6).

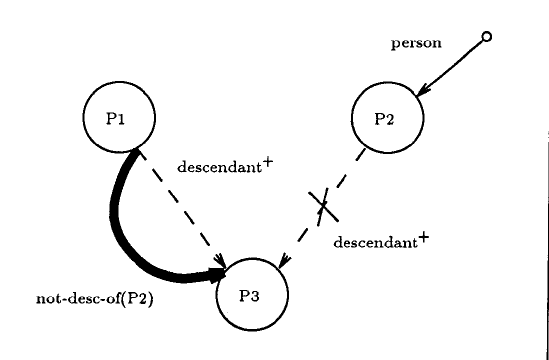


Рисунок 6. Запрос на языке GraphLog

Запрос состоит из трех вершин P1,P2,P3. Пунктирными линиями показаны отношения промаркированные литералами замыкания. Так ребро между P1 и P3 помечено положительным литералом замыкания descendant+, а ребро между P2 и P3 – отрицательным ⌐descendant+. В запросе присутствует еще литерал not-desc-of(P2) и унарный предикат “person”. Данный запрос возвращает тернарный предикат not-desc-of(P1,P3,P2) с потомками P3 “person” P1, которые не являются потомками “person” P2. Код предиката приведен далее:

not-desc-of(P1,P3,P2) ← descendant-tc(P1,P3),

⌐ descendant-tc(P2,P3),

person(P2).

descendant-tc(X,Y) ← descendant(X,Y).

descendant-tc(X,Y) ← descendant(X.Z),

descendant-tc(Z,Y).

## 7.3. G-CORE

Еще одной наиболее свежей и интересной разработкой является язык G-CORE [11]. G-CORE — это графовый язык, который используется для описания и анализа графов. Модель данных G-CORE – модель свойств путей графа (Path Property Graph model) расширяет модели данных графа свойств [14, 15] путями. Благодаря этому G-CORE поддерживает поиск путей и композиционность, позволяет выполнять запросы с указанием атрибутов путей. В отличие от двух предыдущих языков запросы G-CORE задаются синтаксически. Результатом запроса является граф - Path Property Graph model, который может состоять из вершин и ребер, только из ребер, только из путей или быть пустым. Запрос G-CORE содержит описание исходного графа, путей, и может содержать. комбинацию базовых запросов графа с операциями объединения, пересечения и разности.

Основные функции языка G-CORE для обхода графа приведены в ниже.

Самый простой навигационный запрос пути от вершины с меткой Author до вершины с меткой Paper через ребро с меткой worksA на языке G-CORE записывается так:

Данный запрос по необходимости можно усложнять фильтрами по свойствам:

Можно добавлять операции соединения с помощью выражения OPTIONAL:

Самое интересное с точки зрения навигации – возможность выполнять запросы с более детальным уточнением пути. Ниже приведен пример запроса с заданием пути (-/ /-), уточнением его стоимости ( ) и количество путей, которые необходимо найти (3 SHORTEST). Этот пример не исчерпывает всех возможностей по заданию путей в запросах языка G-CORE.

# 8. Описание разрабатываемого навигационного языка

## 8.1. Функциональность

Перед разработкой навигационного языка архиграфовой СУБД, в первую очередь, определим задачи, которые данный язык должен решать:

1. Обработка таких структур данных, как таблицы, графы, текстовые документы, многомерные кубы и поисковые индексы;
2. Работа с архиграфовой БД;
3. Запоминание элементов с данными;
4. Навигация по элементам архиграфовой модели;
5. Навигация с использованием различных условий поиска, циклов, фильтров и сортировки;

На основе перечисленных задач сформируем функции навигационного языка:

1. Возможность подключения обработчиков SQL (для таблиц), MetaCypher (для графов), MDX (для многомерных кубов) и обработчика поисковых индексов (для обработки документов). Запросы данных обработчиков выполняются на серверах Pentaho Mondrian (MDX), Cypher Query Engine (MetaCypher), Apache Drill (SQL) и Index Query Engine (поисковые индексы) соответственно. При подключении обработчика создается процедура, где код пишется на соответствующем языке запросов;
2. Возможность совершать запросы (на получение, изменение, удаление данных) к архиграфовой БД;
3. Сохранение извлеченных данных в переменных соответствующего типа (вершина, ребро, метавершина, метаребро и другие). Также сюда можно добавить возможность создания массивов из этих переменных. Опишем типы переменных. Переменная «вершина» содержит в себе только атрибуты. «Метавершина» содержит в себе «фрагмент метаграфа» (это (мета)ребро или (мета)вершина) и такие атрибуты, как название, тип, адрес данных, поля (если тип = таблица). «Метавершиной» могут быть таблица, индекс, многомерный куб и документ. «Ребро» и «метаребро» содержат ссылки на исходную и конечные вершины, направление, атрибуты. «Метаребро» также содержит «фрагмент метаграфа». «Многомерный куб» – это «метавершина», содержащая следующие атрибуты: название, тип, адрес данных с мерами, объект с полями «название меры: тип меры». «Многомерный куб» также содержит вершины, описывающие оси куба, с такими атрибутами: название оси, тип, адрес данных осей, адрес данных с описанием иерархии.
4. Возможность перехода по вершинам архиграфовой модели и вхождения в эти вершины (в случае, если это метавершины) для последующей их обработки в зависимости от их типа;
5. Возможность создания циклов (for), условий (if/else), фильтрации (filter), сортировки (sort by) при навигации по архиграфовой модели;

## 8.2. Синтаксис

Особенностью разрабатываемого в данной работе навигационного языка является то, что он должен быть гибок и быстр в составлении текста запроса. Поскольку изначально команды поступают в сервис через API от интерфейса пользователя. А затем содержание запроса этих команд должно быть переведено на понятный «язык» для обработчиков запросов, передающих его на исполнителей. В таком случае, синтаксис графового языка G-CORE является наиболее удобным решением для учета данной особенности за счет возможности расширения шагов обхода графа путем «присоединения дополнительного действия к последовательности шагов».

Таким образом, язык G-CORE был выбран в качестве ориентира для разработки навигационного языка. Опишем структуру кода языка. Язык имеет основные типы переменных:

* Attributes (атрибуты). Объект представляет собой неупорядоченный массив, содержащий пары <ключ-значение>. Функции:
* Создание: var attrs = Attributes.create({“name”: “value”});
* Получение элемента: attrs.get(“name”);
* Изменение: attrs.update(“name”, “new\_value”);
* Удаление нескольких элементов: attrs.delete({“name”});
* Добавление нескольких элементов: attrs.add({“name”: “value”});
* Проверка на существование элемента: attrs.has(“name”);
* Получение количества атрибутов: attrs.size().
* V (вершина): объект с полями name (типа string) и attributes (типа Attributes). Создание: var v = V.create(“name”, <attributes>);
* E (ребро): объект с полями startVertex (исходная вершина, тип MV | V), endVertex (конечная вершина, тип MV | V), direction (направление, тип bool, направленное ребро – true, ненаправленное – false), attributes (тип Attributes). Создание: var e = E.create(<startVertex>, <endVertex>, <direction>, <attributes>);
* MV (метавершина): объект, имеющий те же поля, что и V, и, дополнительно, поле data (типа MVE). Создание: var mv = MV.create(“name”, <data>, <attributes>);
* ME (метаребро): объект, имеющий те же поля, что и E, и, дополнительно, поле data (типа MVE). Создание: var me = ME.create(<startVertex>, <endVertex>, <data>, <direction>, <attributes>);
* MVE (фрагмент метаграфа): множество элементов типа <V | E | MV | ME>;
* MetaGraph (метаграф). Объект, состоящий из «фрагментов метаграфа» (вершин, метавершин, ребер, метаребер). Функции:
* Создание: var mg = MetaGraph.create({<MV>}, {<MV1> -> <MV2>});
* Получение массивов элементов: mg.getVertexs(<len>), mg.getMetaVertexs(<len>), mg.getEdges(<len>), mg.getMetaEdges(<len>).

Здесь len – ограничение количества элементов;

* Добавление нескольких элементов: mg.add({<MV>},   
  {<MV1> -> <MV2>});
* Получение количества элементов: mg.size() – всех, mg.VertexsSize(), mg.MetaVertexsSize(), mg.EdgesSize(), mg.MetaEdgesSize().
* Table (таблица): имеет поля data (двумерный массив значений таблицы) и columns (одномерный массив названий столбцов таблицы). Значения массивов – типа string;
* MultiCube (многомерный куб): имеет поля name (string), type (string), measureFields (тип Attributes), measure (ссылка на данные с мерами) и axes (массив элементов типа CubeAxis);
* CubeAxis (ось куба): имеет поля name (string), type (string), values (ссылка на данные осей), hierarchy (ссылка на данные с описанием иерархии);
* Index (поисковый индекс);
* Document (документ);
* AV (вершина архиграфа): объект с полями name (типа string), type (типа string), attributes (типа Attributes) и data (типа AVE).
* AVE (фрагмент архиграфа): множество элементов типа <V | E | MV | ME | Table | MultiCube | Index | Document>.
* ArchGraph (архиграф). Данный объект содержит в себе структуру архиграфа, который обычно возвращается из архиграфовой БД, но также его можно создать самостоятельно. Объект состоит из вершин, метавершин, ребер, метаребер и атрибутов. Функции:
* Создание:   
  var agraph = ArchGraph.create({<AV>},   
  {<AV1> -> <AV2>});
* Остальные функции аналогичны типу MetaGraph, только теперь еще добавились типы Table, MultiCube, Index, Document.

Также в языке присутствуют другие простые типы:

* string (строка);
* int (число);
* bool (логическая переменная, имеет 2 значения: true, false);
* Error (ошибка). Это объект с двумя полями: тип (type) и текст (text) ошибки.

Кроме того, из элементов выше можно создавать массивы. Синтаксис:

<Type>[] elem\_name = []. Пример: E[] edges = []. Функции массивов:

* Добавление: edges.push(<edge>);
* Получение: var edge = edges[1];
* Изменение: edges[2] = edge;
* Удаление: edges.delete(2);

**Базовая навигация.**

Перейдем к описанию базовой навигации. Взятый в основу язык навигации по графу G-CORE было необходимо адаптировать под универсальную модель данных, данные которой имеют 9 различных типов, а не только ребра вершины.

Для этого было решено конкретизировать тип элемента архиграфа, по которому ведется поиск. Переменная находится внутри контейнера с ее типом:

где *ElemType* ⸦ {V | MV | Table | MultiCube | Index | Document};

*variable –* название переменной, участвующей в запросе;

*Label –* метка.

Тогда конструкция MATCH будет иметь вид, как на примере:

Аналогичным образом будет конкретизироваться тип связи между объектами архиграфа:

где *EdgeType* ⸦ { E | ME };

*Label –* метка.

Тогда конструкция MATCH будет иметь вид, как на примере:

Так же необходимо, чтобы результатом выполнения запроса к архиграфовой СУБД было множество переменных различных типов, поскольку при обходе архиграфа на пути от объекта О1 до объекта О2 могут встретиться объекты отличных от них типов. Мы решили, что наиболее простым и удобным вариантом будет хранение результата запроса в виде двумерного массива arr[m][n], то есть массива размера m с массивами размера n с информацией о путях. Массив с информацией о путях содержит последовательность объектов и связей для переходов между ними. Пример:

Результат:

На данный момент мы рассмотрели и адаптировали под универсальную модель данных самые простые выражения языка G-CORE. Поскольку нашей задачей является создание простейшей навигации по архиграфовой модели. Конечно, мы понимаем, что так же необходимо расширить функционал выражения MATCH способностью погружения в содержание метавершины, если она будет присутствовать в теле запроса. Выражение CONSTRUCT с указанием связи так же требует расширения возможностью погружения в метаребро и получением информации из него, если данный тип связи будет указан в качестве пути. Мы решили, что с синтаксической стороны мы не будем создавать какие-либо дополнительные конструкции для указанных ситуаций, и предполагаем решить данную проблему в процессе реализацию данных функций в движке навигационного языка.

# 9. Заключение

Был произведен анализ предметной области, рассмотрены примеры существующих навигационных языков, наиболее подходящих задачи ВКРМ, разработаны функции и синтаксис для навигационного языка собственной архиграфовой СУБД. В результате анализа были сделаны выводы, которые в дальнейшем планируется использовать в ВКРМ.

# 10. Список использованной литературы

1. Basu A., Blanning R. Metagraphs and their applications. Springer, 2007. 174 p. DOI: 10.1007/978-0-387-37234-1.
2. Globa,, L. S., Ternovoy, M. Y., & Shtogrina, O. S. (2015) Metagraph based representation and processing of the fuzzy knowledge bases. Open Semantic Technologies for Intelligent Systems (5) 237-240.
3. Astanin, S. V., Dragnish, N. V., & Zhukovskaya, N. K. (2012) Nested metagraphs as models of complex objects. Engineering journal of Don, 23 (4-2) 76-80.
4. Samokhvalov, E. N., Revunkov, G. I., Gapanyuk, Yu. E. (2015) Metagraphs for Information Systems Semantics and Pragmatics Definition. Herald of the Bauman Moscow State Technical University. Series Instrument Engineering, (1 (100)), 83-99.
5. Tarassov, V. B., & Gapanyuk, Y. E. (2020). Complex Graphs in the Modeling of Multi-agent Systems: From Goal-Resource Networks to Fuzzy Metagraphs. In: Kuznetsov, S. O., Panov, A. I., & Yakovlev, K. S. (eds) Artificial Intelligence RCAI 2020 (pp. 177-198) Lecture Notes in Computer Science, vol 12412. Springer, Cham. DOI: 10.1007/978-3-030-59535-7\_13.
6. Kruchinin, S. V. (2017) On some generalizations of graphs: multigraphs, hypergraphs, metagraphs, flow and port graphs, protographs, archigraphs. Science issues, (3), 48-67. https://elibrary.ru/download/elibrary\_32627955\_99311654.pdf.
7. Kruchinin, S. V. (2017). Protogaphs and Archigraphs as a Graphs Generalization. Journal of Scientific Research Publications, (3 (41)), 23-33. https://elibrary.ru/download/elibrary\_30637766\_78239877.pdf.
8. Sukhobokov, A. A. (2022) Metagraph-tabular data model for asset management systems. In Artificial Intelligence in Management, Control, and Data Processing Systems. Proceedings of the All-Russian Scientific Conference (Moscow, April 27–28, 2022), Publishing house of BMSTU, Moscow (Vol. 1, pp. 93-99).
9. I.A. Erokhin, N.S. Grunin, A.V. Molchanov, E.A. Belousov, Yu.E. Gapanyuk (2022) Мethod of storing metagraph data model in postgresql DBMS. In IIAS'22 Artificial Intelligence in Management, Control, and Data Processing Systems. Proceedings of the All-Russian Scientific, Vol. 2., ISBN 978-5-7038-5910-0., Moscow, Russia.
10. V. Chernenkiy., Y. Gapanyuk and et al, “Storing Metagraph Model in Relational, Document-Oriented, and Graph Databases.”, International Conference on Data Analytics and Management in Data Intensive Domains Proceedings (2018).
11. Sukhobokov A.A., Trufanov V.A., Stolyarov Y.A., Sadykov M.R., Elizarov O.O. Distributed meta graph DBMS based on Blockchain technology //Natural and technical sciences. – 2021. – No. 7. – PP. 201-209.–URL: <https://doi.org/10.25633/ETN.2021.07.15>.
12. Mariano P. Consens and Alberto O. Mendelzon. 1990. GraphLog: a visual formalism for real life recursion. In Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS '90). Association for Computing Machinery, New York, NY, USA, 404–416. <https://doi.org/10.1145/298514.298591>
13. M. Consens and A. Mendelzon. Graphlog: A visual formalism for real life recursion. In 9th ACM Symposium on Principles of Database Systems (PODS), pages 404-416, 1990.
14. Angles R. et al. G-CORE: A core for future graph query languages //Proceedings of the 2018 International Conference on Management of Data. – 2018. – С. 1421-1432.
15. Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of Modern Query Languages for Graph Databases. Comput. Surveys 50, 5 (2017). <https://doi.org/10.1145/3104031>