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 интерфейса системы поддержки мультипарадигмального озера данных, использующего универсальную модель данных

Студент ИУ5-35М _____ **С. В. Очеретная**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **А.В. Сухобоков**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ **А.В. Сухобоков**
(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 интерфейса системы управления мультипарадигмальным озером данных, использующим универсальную модель данных_____

Студент группы __ИУ5-35М

_____Очеретная_Светлана_Вячеславовна_____
(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 интерфейса системы управления мультипарадигмальным озером данных, использующим универсальную модель данных_____*

_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 34 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания « 9 » сентября 2024 г.

Руководитель НИР _____ _____ **А.В. Сухобоков**
(Подпись, дата И.О.Фамилия)

Студент _____ _____ **С.В. Очеретная**
(Подпись, дата И.О.Фамилия)

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

1. Оглавление

2. Введение 4
3. Постановка задачи 4
4. Актуальность 5
5. Описание предметной области 6
   5.1. Общие сведения о метаграфах 6
   5.2. Понятия протографа и архиграфа 8
   5.3. Разработка архиграфой модели 10
   5.4. Хранение универсальной модели данных 11
6. Анализ существующих СУБД 15
   6.1. Virtuoso 15
   6.2. Microsoft SQL Server 17
   6.3. Oracle 18
   6.4. Teradata 20
7. Сравнение разрабатываемой СУБД с аналогами 22
8. Описание разрабатываемого навигационного языка 27
   8.1. Функциональность 27
   8.2. Синтаксис 29
9. Заключение 34
10. Список использованной литературы 34

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: $MG = \langle V, MV, E, ME \rangle$, где MG – метаграф, V – множество вершин метаграфа, MV – множество метавершин метаграфа, E – множество рёбер метаграфа, ME – множество метарёбер метаграфа.

Вершина определяется как: $v = \langle \{atr_1, \cdots, atr_k\} \rangle, v \in V$, где v – вершина метаграфа, $atr_1, \cdots, atr_k$ – атрибуты вершины.

Ребро метаграфа описывается как: $e = \langle v_{begin}, v_{end}, eo, \{atr_1, \cdots, atr_k\} \rangle$, $e \in E \wedge eo \in \{true, false\}$, где e – ребро метаграфа, v_{begin} – исходная вершина (метавершина) ребра, v_{end} – конечная вершина (метавершина) ребра, $atr_1, \cdots, atr_k$ – атрибуты ребра, eo – признак направленности ребра (eo = true – направленное ребро; eo = false – ненаправленное ребро).

Метавершина задаётся как: $mv = \langle EV, \{atr_1, \cdots, atr_k\} \rangle, mv \in MV$, где mv – метавершина метаграфа, $atr_1, \cdots, atr_k$ – атрибуты метавершины, EV – фрагмент метаграфа.

Фрагмент метаграфа в общем виде задаётся как: $EV = \langle \{ev | ev \in (V \cup E \cup MV \cup ME\} \rangle$, где EV – фрагмент метаграфа, ev – элемент, который является либо ребром, либо метаребром, либо вершиной, либо метавершиной.

Метаребро задаётся как: $me = \langle v_{begin}, v_{end}, eo, \{atr_1, \cdots, atr_k\}, EV \rangle$, $me \in ME \wedge eo \in \{true, false\}$, где me – метаребро метаграфа, v_{begin} – исходная вершина (метавершина) ребра, v_{end} – конечная вершина (метавершина) ребра, $atr_1, \cdots, atr_k$ – атрибуты ребра, eo – признак направленности ребра (eo = true – направленное ребро; eo = false – ненаправленное ребро), EV – 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.

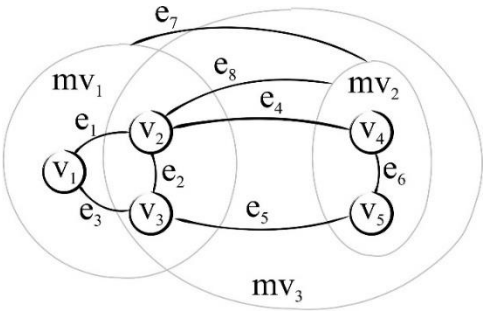


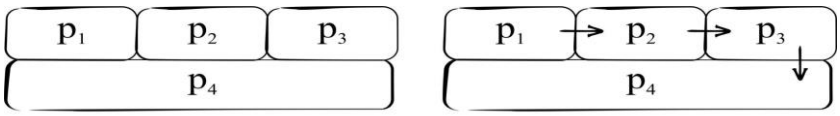
Рис. 1. Пример аннотируемого метаграфа

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: $G^n = \langle V_1, V_2, \cdots, V_n \rangle$, где G^n – архиграф, V_i – множество элементов, n – число множеств. Таким образом, можно сказать, что архиграф состоит из некоторого числа классов, где V_i содержит множество элементов i-го класса. Примером архиграфа 2 степени: $G^2 = \langle V_1, V_2 \rangle$, – является обычный граф, заданный как: $G = \langle E, V \rangle$, где E – множество рёбер, V – множество вершин.

Протографом называется множество элементов $P = \{p_1, p_2, \cdots, p_n\}$ и матрица их соседства: $M = \|m_{i,j}\|_{n \times n}, m_{i,j} \in \{0,1\}$, – в которой 1 означает наличие соседства элемента p_i с элементом p_j, 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.



а) б)

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 G^n можно определить как протограф P, 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.

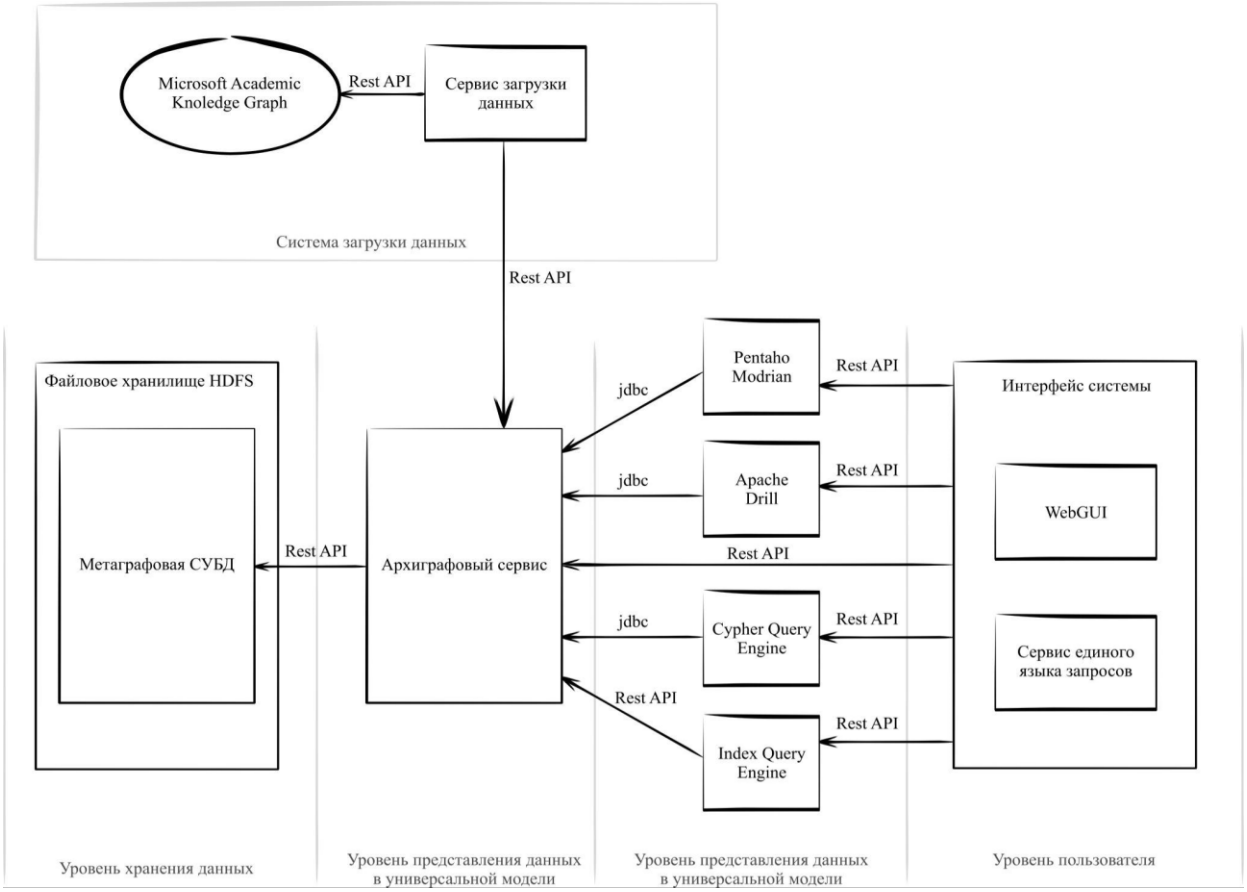


Рисунок 3. Возможная схема озера данных на основе универсальной модели данных

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.

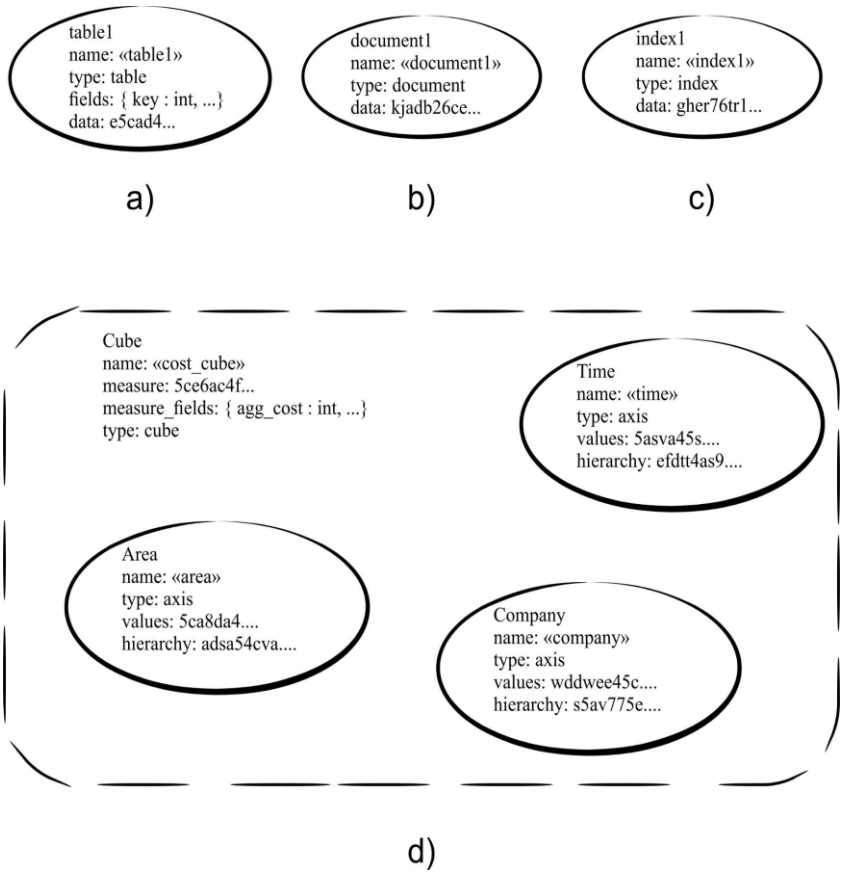


Рисунок 4. Вариант представления в метаграфе классов:
a) таблиц, b) документов, c) индексов, d) многомерных кубов

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. Анализ существующих СУБД

Проведем анализ различных систем управления базами данных, работающих со структурами, с которыми будет взаимодействовать разрабатываемая архиграфовая СУБД с собственным навигационным языком, вызывающим обработчики для разных форматов данных. Будем рассматривать задачу, заключающуюся в обработке большого объема данных, содержащего таблицы, документы, поисковые индексы, графы и многомерные кубы. Для этого необходимо подобрать такие СУБД, которые могут работать с различными структурами данных. Наиболее подходящими для решения поставленной задачи системами являются Virtuoso, Microsoft SQL Server, Oracle, Teradata. Данные примеры СУБД были взяты из списка DB-engines [12].

6.1. Virtuoso

Первой рассматриваемой системой становится Virtuoso – полнофункциональная СУБД, разработанная компанией OpenLink Software. Она предоставляет широкий набор возможностей и поддерживает различные структуры данных. Virtuoso поддерживает следующие структуры данных:

– Реляционные таблицы: Virtuoso поддерживает реляционные таблицы и предоставляет возможности для создания, модификации и запросов к таблицам;

– Графовые данные: Virtuoso разработана как гибридная база данных и поддерживает RDF-графы, графы RDF*, графы SPARQL, графы OWL и другие графовые структуры данных. Это позволяет эффективно хранить и обрабатывать графовые структуры данных;

– Документы: Virtuoso обладает встроенной поддержкой XML и может обрабатывать и хранить XML-документы;

– Полнотекстовый поиск: СУБД Virtuoso имеет возможность индексировать и искать данные по полнотекстовому поиску;

– Многомерные кубы: Virtuoso поддерживает аналитическую обработку данных с помощью многомерного моделирования и агрегации.

Приведем преимущества Virtuoso:

– Мощность и гибкость: система способна обрабатывать большие объемы данных и обеспечивать высокую производительность в сочетании с масштабируемостью;

– Поддержка разных типов данных: СУБД предлагает широкий набор функций для работы с различными структурами данных, что позволяет разработчикам выбрать подходящий формат для своих данных;

– Совместимость с различными стандартами: Virtuoso поддерживает стандарты RDF, SPARQL, SQL, XML и другие, что обеспечивает совместимость и интеграцию с другими системами и инструментами.

Далее приведем недостатки Virtuoso:

– Сложность: Настройка и использование Virtuoso может быть сложной задачей для пользователей без опыта работы с графовыми и RDF-структурами данных;

– Стоимость: система лицензируется как коммерческое ПО, и его использование может потребовать дополнительных затрат на лицензию;

– Ограниченная экосистема: СУБД имеет меньшую экосистему и сообщество пользователей по сравнению с некоторыми другими СУБД, что может повлиять на доступность поддержки и ресурсов.

В целом, Virtuoso – это мощная СУБД, которая обладает широкими возможностями и поддерживает различные структуры данных, но требует некоторых знаний и вложений для эффективного использования.

6.2. Microsoft SQL Server

Далее рассмотрим реляционную СУБД Microsoft SQL Server, разработанную корпорацией Microsoft, которая может обрабатывать следующие структуры данных:

– Реляционные таблицы: MS SQL Server использует модель реляционных таблиц для хранения данных. Он предоставляет возможность создания, модификации и запросов к таблицам, а также поддерживает различные типы данных, такие как числа, строки, даты и другие;

– Графовые данные: В последних версиях MS SQL Server (начиная с 2017 года) была введена поддержка графовых данных с помощью реализации языка запросов Cypher от Neo4j. Это позволяет хранить и обрабатывать данные в виде графа;

– Документы: MS SQL Server обладает встроенной поддержкой хранения и обработки XML-документов с использованием дополнительных типов данных и функций.

– Многомерные кубы: Многомерная обработка в данной СУБД осуществляется с помощью SQL Server Analysis Services – эти службы устанавливаются дополнительно к системе;

– Поисковые индексы: в систему также включен полнотекстовый поиск [13], однако изначально СУБД не предназначена для выгодного использования поисковых индексов.

Перечислим преимущества MS SQL Server:

– Широкая функциональность: MS SQL Server предлагает широкий набор функций и возможностей для управления, хранения и обработки данных. Он поддерживает транзакционность, процедуры, триггеры, представления, главные диаграммы и другие функции, которые обеспечивают гибкость и мощность системы;

– Интеграция с другими продуктами Microsoft: MS SQL Server является частью экосистемы Microsoft и интегрируется с другими продуктами, такими как

Microsoft Azure, Power BI, SharePoint и другими. Это обеспечивает простую интеграцию данных и возможность создания полного стека приложений на платформе Microsoft;

– Масштабируемость и производительность: MS SQL Server поддерживает масштабирование, благодаря чему может обрабатывать большие объемы данных. Он обладает также оптимизатором запросов, индексацией и другими механизмами, которые улучшают производительность запросов и обработку данных.

Далее приведем недостатки MS SQL Server:

– Платформа Windows: MS SQL Server является проприетарной СУБД от Microsoft и в первую очередь ориентирован на платформу Windows. В результате, для его использования может понадобиться операционная система Windows Server.

– Лицензирование: MS SQL Server требует лицензирования и может быть дорогим в использовании. Стоимость лицензий и поддержки может быть значительна для больших предприятий или проектов.

– Сложность настройки и администрирования: из-за широкого набора функций и настроек, MS SQL Server может требовать дополнительных усилий и знаний для эффективной настройки и администрирования.

Таким образом, MS SQL Server является зрелой и мощной реляционной СУБД с широким набором функций и возможностей, но требующей некоторых затрат и экспертизы для эффективного использования. Система подходит для различных типов приложений, от малых до крупномасштабных предприятий.

6.3. Oracle

Следующей приведенной в данной работе системой становится мощная, широко используемая реляционная СУБД Oracle, разработанная Oracle Corporation. Рассмотрим структуры данных, поддерживаемые системой:

– Реляционные таблицы: Oracle работает с реляционными таблицами, предоставляя возможности для создания, модификации и запросов к данным. Он поддерживает различные типы данных и операции, такие как соединение таблиц, группировку, сортировку и другие;

– Графовые данные: с начала 2021 года Oracle начала поддерживать графовые данные с помощью Oracle Property Graph (PGX) - графовой аналитической технологии Oracle. PGX обеспечивает возможности для хранения и обработки графовых структур данных;

– Документы: Oracle предоставляет встроенную поддержку для работы с XML-документами, включая функции для разбора, генерации и обработки XML-данных.

– Поисковые индексы: Oracle также поддерживает поисковые индексы для оптимизации поиска и полнотекстовых запросов, однако это свойство не является основным в системе;

– Многомерные кубы: Oracle поддерживает многомерные кубы [14] с помощью своего инструмента Oracle OLAP (Online Analytical Processing).

Приведем преимущества Oracle:

– Масштабируемость и производительность: Oracle обеспечивает высокую производительность при обработке больших объемов данных и масштабируется с ростом потребностей. Он имеет механизмы для оптимизации запросов, параллельной обработки и управления ресурсами, что позволяет достичь хорошей производительности;

– Надежность и безопасность: Oracle предлагает высокий уровень надежности и устойчивости, что является особенно важным для критических систем. Он также обеспечивает механизмы безопасности, включая управление доступом, шифрование и аудит;

– Широкий набор функций: Oracle предлагает богатый набор функций и возможностей, включая хранение процедур, триггеров, представлений, геопространственных данных, аналитические функции, OLAP, аудит и др. Это обеспечивает гибкость и мощность при разработке приложений и запросов.

Перечислим недостатки Oracle:

– Цена: Лицензирование Oracle может быть довольно дорогим, особенно для больших предприятий или проектов. Возможно, требуется расширенное понимание и определенные ресурсы для эффективного использования.

– Сложность: Oracle является мощной СУБД, но из-за широкого набора функций и настроек, его использование может требовать серьезного знания и опыта администрирования.

– Зависимость от экосистемы Oracle: Oracle предлагает свою экосистему инструментов и поддержки, но интеграция с другими системами или ПО может оказаться более сложной, особенно если они используют другие СУБД или технологии.

– Потребление ресурсов: Oracle может потреблять значительное количество ресурсов, включая память, процессор и дисковое пространство, особенно при работе с большими объемами данных.

В итоге следует сказать, что Oracle является одной из наиболее популярных и мощных систем управления базами данных на рынке. Она обладает широким функционалом, высокой производительностью и надежностью, что делает ее идеальным выбором для различных приложений и предприятий. Однако работа с этой системой требует затрат, знаний и вычислительных ресурсов.

6.4. Teradata

Последней рассматриваемой мультимодальной СУБД является Teradata, предназначенная для обработки больших объемов данных. Она может работать с различными структурами данных, включая:

– Таблицы: традиционные структурированные данные хранятся в виде таблиц, состоящих из строк и столбцов;

– Графы: Teradata поддерживает модель графовой базы данных, где данные представляются в виде узлов и ребер, что облегчает анализ связей и отношений между различными элементами;

– Документы: Teradata может работать с неструктурированными данными в формате документов, такими как JSON и XML, что позволяет хранить и извлекать такие данные с помощью соответствующих операций и индексов;

– Поисковые индексы: мощные возможности индексирования и поиска позволяют Teradata эффективно обрабатывать запросы по текстовым данным;

– Многомерные кубы: Teradata предоставляет функциональность анализа многомерных данных, включая возможность создания, хранения и обработки многомерных кубов.

Teradata имеет следующие преимущества:

– Масштабируемость: Teradata способна масштабироваться горизонтально и вертикально, обрабатывая большие объемы данных и поддерживая распределение данных между несколькими серверами;

– Высокая производительность: Оптимизированная обработка запросов и параллельное выполнение операций позволяют достичь высоких скоростей работы с данными;

– Гибкость аналитики: Teradata предоставляет мощные инструменты для аналитики данных, позволяющие выполнять сложные аналитические задачи, включая OLAP-запросы и многомерный анализ;

– Надежность: Система обладает высокой отказоустойчивостью и предоставляет механизмы резервирования и восстановления данных.

Teradata также имеет свои недостатки:

– Высокая стоимость: запуск и поддержка системы Teradata может быть дорогостоящей, особенно для малых и средних предприятий;

– Сложность настройки: Teradata требует глубокого понимания и опыта в настройке и управлении базой данных, что может быть сложным для новичков;

– Ограниченная экосистема: в отличие от некоторых других более популярных СУБД, Teradata имеет ограниченное количество инструментов и расширений, доступных для работы;

– Сложность миграции: Перенос существующих данных из другой СУБД в Teradata может потребовать дополнительных усилий и времени из-за различий в структуре данных и синтаксисе запросов.

Таким образом, Teradata является мощной СУБД, способной эффективно обрабатывать и анализировать большие объемы данных, и подходит для предприятий, которым требуется высокая производительность и масштабируемость. Однако использование Teradata может быть дорогостоящим и требовать опыта в настройке и управлении базой данных. Кроме того, экосистема Teradata может оказаться ограниченной, и миграция данных из другой СУБД может потребовать дополнительных усилий.

7. Сравнение разрабатываемой СУБД с аналогами

Сравним разрабатываемую архиграфовую СУБД с рассмотренными выше аналогами, используя метод взвешенной суммы. Сначала расставим оценки важности для всех критериев, по которым сравниваем системы. Будем использовать оценки от 1 до 10. (1 – менее значимый, 10 – самый значимый). Данные оценки приведены в таблице 1.

Таблица 1. Оценки критериев

Название критерия / оценка важности	**Менее значимый (1)**	**Более значимый (10)**
Цена	дорого	дешево
Функциональность	мало функций	большое число функций
Поддержка различных структур данных	поддерживает малое количество структур	поддерживает огромное количество структур
Мощность	низкая	высокая

Название критерия / оценка важности	**Менее значимый (1)**	**Более значимый (10)**
Сложность	высокая	низкая
Интеграция с другими системами	сложная	легкая
Надежность	низкая	высокая
Зависимость от ОС	высокая	отсутствует
Поддержка	не развитое сообщество	развитое сообщество

В таблице 2 представлены оценки критериев для каждого варианта СУБД.

Таблица 2. Значения критериев для вариантов СУБД

Критерии		**Варианты**				
		B1	**B2**	**B3**	**B4**	**B5**
		Virtuoso	**MS SQL Server**	**Oracle**	**Teradata**	**Архиграфовая**
К1	Цена, за год	9	7	1	6	10
К2	Функциональность	7	10	10	3	5
К3	Поддержка различных структур данных	8	5	6	6	9
К4	Мощность	6	7	9	8	7
К5	Сложность	5	3	3	5	9
К6	Интеграция с другими системами	8	6	3	2	5
К7	Надежность	6	6	9	8	6
К8	Зависимость от ОС	9	2	9	5	4
К9	Поддержка	3	9	10	7	2

Поясним значения критериев. При сравнении будем также использовать источники [15, 16, 17, 18]. Сначала поясним значения цен, которые были

выставлены по шкале, представленной в таблице 3. Наиболее высокую цену (около 1млн/год) имеет Oracle, соответственно для нее стоит оценка 1. Наиболее низкую цену (100к/год) имеет Virtuoso, для которого стоит оценка 9. Наша система разрабатывается бесплатно и имеет оценку 10. MS SQL Server имеет цену около 330к/год и оценку 7. Teradata имеет цену около 436к/год и оценку 6.

Таблица 3. Шкала цен.

Интервал цены, тыс.руб/год	бесплатно	<125	<250	<375	<500	<625	<750	<875	<1125	<1125
Оценка	10	9	8	7	6	5	4	3	2	1

Наибольшей функциональностью обладают MS SQL Server и Oracle, поэтому для них стоят оценки 10. Система Virtuoso также обладает хорошей функциональностью, но более низкой, чем у предыдущих двух СУБД, поэтому для нее ставим оценку 7. Teradata имеет мало функций и имеет оценку 3. Нашей системе сейчас сложность дать оценку функциональности, но планируется сделать оптимальное количество функций, достойное средней оценки 5.

Наибольшее число структур (все необходимые) поддерживает Virtuoso и имеет оценку 8. Наша СУБД будет поддерживать много различных структур путем внедрения различных обработчиков и имеет оценку 9. Oracle, MS SQL и Teradata Server первично поддерживают обработку таблиц, вторично – графов, документов, пространственных объектов. Oracle также вторично является хранилищем RDF. Teradata также используется для хранения данных, основанных на времени. Все остальные структуры эти 3 СУБД поддерживают на основе установленных дополнительно инструментов. Исходя из сказанного, Oracle и Teradata имеют оценку 6, а MS SQL Server – 5.

Все СУБД являются достаточно мощными, однако Oracle обладает наиболее высокой производительностью, затем Teradata, затем MS SQL Server, затем Virtuoso. На основе этого поставим оценки от 6 до 9 в соответствии с ранжировкой.

Из-за большого числа функций Oracle и MS SQL Server обладают наиболее высокой сложностью (настройки и администрирования), поэтому имеют оценки 3. Teradata и Virtuoso проще в использовании, но также достаточно сложны, в связи с этим имеют оценки 5. Наша система самая простая, так как находится на этапе развития, и имеет оценку 9.

Системы Oracle и Teradata сложно интегрировать с другими. При этом Oracle можно легко интегрировать только с инструментами ее экосистемы. На основе этого Oracle имеет оценку 3, Teradata оценку 2. Virtuoso поддерживает различные стандарты и легко интегрируется с другими системами, в связи с чем имеет оценку 8. MS SQL Server может интегрироваться с большим количеством продуктов экосистемы Microsoft, на основе чего имеет оценку 6. Наша система поддерживает различные структуры, что упрощает интеграцию, в связи с этим ей можно поставить оценку 5.

Наиболее высокой надежностью обладает Oracle, имеющая собственные механизмы защиты, и имеет оценку 9. Также механизмы отказоустойчивости, восстановления данных и резервирования хорошо настроены в Teradata, которой поставим оценку 8. Остальным СУБД поставим оценки 6.

Microsoft SQL Server поддерживает только на операционных системах Windows, в связи с чем имеет оценку 2. Oracle и Virtuoso поддерживаются на Windows, Linux, Unix и имеют оценку 9. Teradata поддерживается только на некоторых ОС: MP-RAS UNIX, Microsoft Windows 2000/2003 Server, SuSE Linux. В связи с этим поставим для Teradata оценку 5. Наша система будет работать на ОС семейства Linux, в связи с этим имеет оценку 4.

Самой популярной является СУБД Oracle и имеет оценку 10. Далее идет MS SQL Server с оценкой 9, далее Teradata с оценкой 7 и совсем небольшим сообществом обладает Virtuoso, в связи с чем имеет оценку 3. Наша система также пока не имеет хорошей поддержки и имеет оценку 2.

Далее вычислим нормированные значения критериев по формуле 1:

$$k_{ij} = \frac{X_{ij}}{X_i^+} \quad (1)$$

где $X_i^+ = \max_j X_{ij}$ – значение i-го локального критерия, соответствующее максимальному значению среди сравниваемых вариантов решения.

Далее расставим критерии важности (параметр а). Наиболее важным будем является критерий К3 (а = 4/20), на втором месте критерии К1, К2 (а = 3/20), на третьем – К4-К7 (а = 2/20), на последнем К8, К9 (а = 1/20). При этом сумма всех критериев равняется 1.

Нормированные значения критериев и критерии важности запишем в таблицу 4.

Таблица 4. Нормированные значения критериев и критерии важности.

Критерии	**Варианты**					**коэффициенты важности**
	В1	**В2**	**В3**	**В4**	**В5**	
К1	0,9	0,7	0,1	0,6	1	0,15
К2	0,7	1	1	0,3	0,5	0,15
К3	0,888889	0,55556	0,6667	0,6667	1	0,20
К4	0,666667	0,77778	1	0,8889	0,7778	0,10
К5	0,555556	0,33333	0,3333	0,5556	1	0,10
К6	1	0,75	0,375	0,25	0,625	0,10
К7	0,666667	0,66667	1	0,8889	0,6667	0,10
К8	1	0,22222	1	0,5556	0,4444	0,05
К9	0,3	0,9	1	0,7	0,2	0,05

Вычислим взвешенную сумму по формуле 2:

$$Y_l = \sum_i^n \alpha_i \, k_{ij} \quad (2)$$

где n — число критериев сравнения;

l – номер сравниваемого варианта;

α_i — коэффициент важности i-го параметра сравнения;

k_{ij} — коэффициент нормализации, определяет уровень соответствия i-го параметра j-го варианта наилучшему значению.

Вычисленные значения взвешенных сумм для каждого представлены в таблице 5, где видно, что наш вариант (B5) является одним из лучших.

Таблица 5. Значения взвешенных сумм.

B1	**B2**	**B3**	**B4**	**B5**
0,7717	0,675	0,669167	0,58944444	0,7642

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

В первую очередь, необходимо описать структуру кода языка. В самом начале кода подключаются обработчики и дополнительные пакеты с помощью оператора include. Затем идет объявление глобальных переменных. И уже затем идет основной блок исполняемого кода, начинающийся с оператора start и заканчивающийся оператором end.

Перед описанием синтаксиса языка следует отметить следующее:

– При описании синтаксиса используются фигурные скобочки {}, означающие то, что элементы внутри них составляют перечисление. То есть данная конструкция {elem} эквивалентна данной: elem, elem, elem … При этом, данное условие не распространяется на конструкции операторов языка.

– В кавычках “” обозначаются текстовые значения, а в скобочках «» - конструкции или значения других типов (чаще всего объектов).

– Переменные могут быть объявлены с помощью ключевого слова var или с явным определением типа. Например, так: var agraph; или так: ArchGraph agraph;

Опишем синтаксис в соответствии с перечисленными в п. 8.1. функциями:

1. Обработчики. Подключение обработчиков производится с помощью оператора «include» следующим образом: include «название_обработчика». Примеры подключения обработчиков:

include «mdx-handler»;

include «sql-handler»;

include «metacypher-handler»;

include «index-handler»;

После подключения обработчиков станут доступны соответствующие объекты: sql, mdx, metacypher, index.

2. **Работа с БД.** За работу с архиграфовой БД отвечает глобальный объект «BD», позволяющий подключиться в базе данных: BD.connect(“название_БД”, “хост”, “порт”, “имя_пользователя”, “пароль”). Пример: BD.connect(“svetlana-db”, “192.168.0.1”, “1234”, “admin”, “123”).

Функция BD.connect(…) возвращает объект типа Database. Пример получения переменной: var database = BD.connect(…). Объекты типа Database имеют следующие функции:

– возможность закрыть соединение: database.close();

– возможность совершить запрос: database.Query(“запрос”). Поскольку подключение производится к архиграфовой БД, то запросы совершаются в соответствии с синтаксисом реляционного языка, который разрабатывается в рамках отдельного подпроекта. Данный запрос может вернуть объект типа ArchGraph, ошибку Error или ничего (пустой объект).

– возможность совершить запрос к специфическим структурам архиграфовой БД. Чтобы совершить специфический запрос с помощью обработчиков нужно использовать перегруженную функцию Query: database.Query(“запрос_на_нужном_языке”, <обработчик>). В качестве обработчика передается соответствующий объект: sql, mdx, metacypher, index. Пример: database.Query(“SELECT * from table;”, sql); Функция Query возвращает тип объекта в зависимости от передаваемого обработчика: для sql – Table, для mdx – MultiCube, для поискового индекса – Index, для документа – Document, для метаграфа – MetaGraph (не архиграфа!).

3. **Переменные.** 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);

4. **Базовая навигация.** Навигация по элементам архиграфовой модели производится с помощью операторов «on» и «to». Навигация производится с целью обработать полученную структуру архиграфа (ArchGraph) и получить из

нее нужные данные. Обработка производится с помощью специальной функции Nav со следующим синтаксисом: Nav(<ArchGraph>) { … return <data>; }. Внутри данной функции по умолчанию создается объект CurVertex (текущая вершина, равная изначально самой первой из объекта ArchGraph). Тип этого объекта: <V | E | MV | ME | Table | MultiCube | Index | Document>. CurVertex всегда можно получить внутри функции Nav. Также внутри Nav можно использовать следующие операторы: To <name>; On <name>. При использовании этих операторов значение CurVertex меняется. С помощью оператора To можно из текущей вершины перейти к другой по имени, а с помощью оператора On можно получить данные вершины (объекту CurVertex присваивается значение из поля data, если оно есть).

5. **Дополнительные операторы навигации:**

– for, while, do while: отвечают за создание циклов. Синтаксис:

- for (объявление_переменной; условие_выхода_из_цикла; итерация) {<тело_цикла>};
  Пример: for (var i = 0; i < 10; i++) { … };
- while (<условие_выхода_из_цикла>) {<тело_цикла>}.
  Пример: while (i < 10) { … };
- do {<тело_цикла>} while (<условие_выхода_из_цикла>).
  Пример: do {…} while (i<10). В отличие от предыдущего оператора, здесь первая итерация цикла выполняется в любом случае;

– if, else: отвечают за создание условий. Синтаксис: if (<условие>) {<выполняемый_код_при_истинном_условии>} else {<выполняемый_код_при_ложном_условии>};

– sort: отвечает за сортировку объектов. Синтаксис: sort(<array>, <field>, <is_asc>). Пример: mvArray = sort(mvArray, “name”, true) – сортировка метавершин по названию;

– filter: отвечает за фильтрацию объектов. Синтаксис: filter(<array>, <func(elem){return <bool>}>). Пример: avArray = filter(avArray, GetOnlyTableFunc) – получить только те вершины архиграфа, которые являются

таблицей. Синтаксис функции фильтрации: Func GetOnlyTableFunc(avElem: AV) { if avElem.type == “table” { return true;} else { return false; } };

– дополнительно: в языке также есть операторы сравнения равно (==), больше (>), меньше (<) и функции: Func func_name({<param>: <type>}) { return <elem>; }.

9. Заключение

Был произведен анализ предметной области, рассмотрены примеры СУБД, поддерживающие несколько разных структур данных, произведен их анализ, 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) Method 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. DB-Engines Ranking. URL: https://db-engines.com/en/ranking/. (дата обращения 28.11.2024).

13. Основы работы с MSSQL. URL: https://www.nic.ru/help/osnovy-raboty-s-mssql_11487.html. (дата обращения 28.11.2024).

14. Н.А.Боярский, А.В.Шовкун. Построение аналитической части корпоративной информационно-аналитической системы средствами Oracle

OLAP Option и BI Beans. URL: http://www.olap.ru/home.asp?artId=481. (дата обращения 28.11.2024).

15. Различия между Oracle и MS SQL: сравнение баз данных. URL: https://uchet-jkh.ru/i/razliciya-mezdu-oracle-i-ms-sql-sravnenie-baz-dannyx/. (дата обращения 28.11.2024).

16. Что такое Teradata? Её отличия от аналогов. URL: https://vc.ru/dev/111923-chto-takoe-teradata-ee-otlichiya-ot-analogov. (дата обращения 28.11.2024).

17. Oracle vs Teradata (How to Migrate in 5 Easy Steps). URL: https://bryteflow.com/oracle-vs-teradata-how-to-migrate-5-easy-steps/. (дата обращения 28.11.2024).

18. Феоктистов Е.Ф. Сравнительный анализ СУБД для туристической социальной сети // Программные продукты и системы. 2020. Т. 33. № 4. С. 714-719.