**АННОТАЦИЯ**

В выпускной квалификационной работе разработана графовая база данных для системы управления проектами с целью интеграции данных из существующих систем управления проектами. Использование разработанной графовой базы данных сократит время отклика на запросы пользователей интегрированной системы и предоставит гибкую модель данных.

В пояснительной записке представлены результаты анализа предметной области, разработки графовой базы данных, реализации скриптов для ее заполнения и создания запросов для извлечения необходимой информации из базы данных.

Выпускная квалификационная работа содержит 14 таблиц, 26 иллюстраций, 2 приложения, ссылается на 28 источников и состоит из 99 страниц.

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 7](#_Toc483951468)

[ВВЕДЕНИЕ 8](#_Toc483951469)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc483951470)

[1.1 Проект и управление проектом 9](#_Toc483951471)

[1.2 Жизненный цикл проекта 10](#_Toc483951472)

[1.3 Системы управления проектами 13](#_Toc483951473)

[1.4 SQL и NoSQL решения 17](#_Toc483951474)

[1.5 Графовые СУБД 22](#_Toc483951475)

[1.5.1 Механизмы вычисления графов 25](#_Toc483951476)

[1.5.2 Преимущества графовых баз данных 27](#_Toc483951477)

[1.5.3 Графовая СУБД Neo4j 29](#_Toc483951478)

[2 ИСПЛЬЗУЕМЫЕ ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ 33](#_Toc483951479)

[2.1 Сравнение MySQL и Neo4j 33](#_Toc483951480)

[2.2 Язык запросов Cypher 36](#_Toc483951481)

[3 РЕАЛИЗАЦИЯ 44](#_Toc483951482)

[3.1 Разработка графовой базы данных на СУБД Neo4j 44](#_Toc483951483)

[3.2 Создание скриптов для заполнения графовой БД 47](#_Toc483951484)

[3.3 Создание запросов на языке Cypher 50](#_Toc483951485)

[3.3.1 Запрос 1. Получение сведений о проектах 50](#_Toc483951486)

[3.3.2 Запрос 2. Получение описания задачи 52](#_Toc483951487)

[3.3.3 Запрос 3. Получение истории изменения задачи 53](#_Toc483951488)

[3.3.4 Запрос 4. Добавление задачи к проекту 55](#_Toc483951489)

[4 ОХРАНА ТРУДА 57](#_Toc483951490)

[4.1 Оценка тяжести труда системного администратора 58](#_Toc483951491)

[4.2 Оценка напряженности труда системного администратора 64](#_Toc483951492)

[4.3 Выводы по разделу 66](#_Toc483951493)

[5 ОБОСНОВАНИЕ ЭКНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ 68](#_Toc483951494)

[5.1 План проведения работ 68](#_Toc483951495)

[5.2 Смета затрат на выпускную квалификационную работу 69](#_Toc483951496)

[5.2.1 Материалы, покупные изделия и полуфабрикаты 69](#_Toc483951497)

[5.2.2 Связь, источники информации 70](#_Toc483951498)

[5.2.3 Расходы на электроэнергию 71](#_Toc483951499)

[5.2.4 Расходы на оплату труда 72](#_Toc483951500)

[5.2.5 Расходы на амортизацию оборудования 73](#_Toc483951501)

[5.3 Стоимость выполнения ВКР 74](#_Toc483951502)

[5.4 Оценка эффективности разработки 75](#_Toc483951503)

[5.4.1 Расчет экономии времени 75](#_Toc483951504)

[5.4.2 Расчет экономии средств 76](#_Toc483951505)

[5.4.3 Расчет окупаемости разработки 76](#_Toc483951506)

[5.5 Выводы по разделу 77](#_Toc483951507)

[ЗАКЛЮЧЕНИЕ 78](#_Toc483951508)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 79](#_Toc483951509)

[ПРИЛОЖЕНИЕ А 82](#_Toc483951510)

[ПРИЛОЖЕНИЕ Б 97](#_Toc483951511)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

ВКР – выпускная квалификационная работа

ЖЦП – жизненный цикл проекта

ПО – программное обеспечение

ЖЦ – жизненный цикл

ТЭО – технико-экономическое обоснование

БД – база данных

СУБД – система управления базами данных

ЭВМ – электронно-вычислительная машина

СУП – система управления проектами

OLTP – Online Transaction Processing – обработка транзакций в реальном времени

RDF – Resource Description Framework – среда описания ресурса

MapReduce – модель распределенных вычислений

NoSQL – Not only SQL – не только SQL

SQL – Structured Query Language – язык структурированных запросов

OLAP – Online Analytical Processing – интерактивная аналитическая обработка

ACID – Atomicity, Consistency, Isolation, Durability – набор свойств гарантирующих надежную работу транзакций: атомарность, согласованность, изолированность, долговечность

URL – Uniform Resource Locator – единообразный локатор ресурса

API – Application Programming Interface – интерфейс прикладного программирования

**ВВЕДЕНИЕ**

В области информационных систем и технологий, стремительно развиваются системы управления проектами. Каждая из существующих систем предлагает свой набор инструментов и сервисов для удобной и эффективной работы над проектом.

В рамках одного проекта, зачастую используется несколько систем управления проектами, из-за чего становится актуальной задача их интеграции.

В настоящей выпускной квалификационной работе осуществляется разработка графовой базы данных для системы управления проектами с целью интеграции данных из существующих систем управления проектами.

Разработка графовой базы данных увеличит показатели производительности, реализует ссылки на записи в существующих системах управления проектами, а также благодаря своей гибкости, даст возможность изменять модель данных или необходимую ее часть на лету, исходя из требований к интегрированной системе. Предоставляемая производительность разработанной графовой БД существенно снизит время отклика БД на запросы пользователей интегрированной системы, что позволит сократить расход материальных средств на обслуживание и мониторинг базы данных.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**
   1. **Проект и управление проектом**

**Определение проекта**

Существует ряд определений понятия “проект”, каждое из которых имеет право на существование.

В США, Институте управления проектами, проект трактуется как “некоторое предприятие с изначально установленными целями, достижение которых определяет завершение проекта”. В английской ассоциации проект менеджеров: “проект – это отдельное предприятие с определенными целями, часто включающими требования по времени, стоимости и качеству достигаемых результатов”. По “Оперативному руководству” №2.20 мирового банка, проект определяется как “комплекс взаимосвязанных мероприятий, предназначенных для достижения, в течение заданного периода времени и при установленном бюджете, поставленных задач с четко определенными целями”.

На основании определений, приведенных выше, можно выделить общие признаки проекта:

* изменения – основное содержание проекта;
* проект не может длиться бесконечно, то есть, ограничен во времени;
* ограниченность в необходимых ресурсах;
* новизна;
* комплексность;
* правовое и организационное обеспечение проекта;
* границы с другими видами деятельности и намерениями [1].

В общем понимании, проект – нечто уникальное, новое, имеющее определенный набор задач и четкие границы, ограничен во времени и ресурсах. Проект уникален тем, что это не обычный процесс, а набор конкретных действий, направленных на достижение поставленной цели.

**Управление проектом**

Управление проектом – это применение знаний, навыков, инструментов и методов для проектной деятельности в соответствии с требованиями проекта. Процесс управления проектом разделяется на пять этапов:

1. инициирование;
2. планирование;
3. выполнение;
4. мониторинг и контроль;
5. закрытие.

Процесс управления проектом тесно связан с жизненным циклом проекта, который будет описан немного позднее. Помимо процесса, управление проектом базируется на знаниях, которые подразделяются на следующие области:

1. интеграция;
2. сфера;
3. время;
4. стоимость;
5. качество;
6. закупка;
7. кадровые ресурсы;
8. связь;
9. управление рисками;
10. управление заинтересованными сторонами.

Обладая всеми вышеперечисленными знаниями, а также следуя этапам процесса управления, можно эффективно управлять проектом и успешно завершать его.

* 1. **Жизненный цикл проекта**

Проект имеет свое начало и свой конец, а также периоды роста, стабильности и спада. В простейшем случае ЖЦП включает в себя начальный, промежуточный и заключительный этапы, фазы [2].

Фазы, этапы жизненного цикла проекта включают в себя выполнение основных мероприятий по проекту:

* разработка ТЭО и рабочего проекта;
* контрактная деятельность;
* организация и финансирование работ;
* создание новых технологий;
* планирование ресурсов и хода работ;
* закупка материалов и оборудования;
* выполнение работ и сдача готовых объектов.

Таким образом, исходя из основных мероприятий по проекту, строятся фазы ЖЦП (рис. 1.1).



Рисунок 1.1 – Основные фазы жизненного цикла проекта

Фазы проекта – это отдельные части в рамках проекта, требующие дополнительного контроля для эффективного получения основного результата проекта, они обычно выполняются последовательно, но в некоторых проектных ситуациях могут перекрываться. Высокоуровневый характер фаз проекта превращает их в элемент жизненного цикла проекта. Фаза проекта не является группой процесса управления проектом.

Структура фаз позволяет разделить проект на логические подгруппы для более легкого управления, планирования и контроля. Количество фаз, необходимость в них и степень налагаемого контроля зависит от размера фаз, сложности и потенциального влияния на проект [3].

Один из важных моментов, характеризующий ЖЦП, является нарастание трудозатрат по фазам жизненного цикла. Типичный вид трудозатрат показан на рисунке 1.2. Условно, на каждую фазу дается одинаковое количество времени [4].

Рисунок 1.2 – Диаграмма распределение трудозатрат по фазам ЖЦ

Проект в основном начинается с идеи, которая появляется у одного человека. Постепенно, по мере формулирования, анализа и оценки этой идеи, привлекаются дополнительные специалисты. Еще больше участников требуется на фазе разработки проекта. Пик трудозатрат приходится на фазу реализации проекта. На последнем этапе происходит постепенное высвобождение участников проектной команды. Следует помнить, что проект должен иметь четкое окончание во времени, после которого все работы по проекту закрываются, и на проект перестают тратиться ресурсы [5].

* 1. **Системы управления проектами**

Система управления проектами – это совокупность инструментов или методов, которые используются для управления проектами. Система может быть, как формальной, так и неформальной, и помогает менеджеру проекта эффективно управлять и завершать его. СУП – это ряд процессов и связанных с ними функций контроля, объединённых в целостную структуру.

СУП строится на основе плана управления проектом, который описывает то, как будет использоваться система. Содержание СУП изменяется в зависимости от области приложения, особенностей организации, сложности проекта и доступности необходимых ресурсов. Система строится так, чтобы максимально соответствовать стратегическим целям и производственным ресурсам клиентской организации [6].

Согласно исходному перечню систем управления проектами, в текущей работе рассматриваются такие системы как:

* GitHub;
* Bitbucket;
* Jira;
* GitLab.

GitHub – это крупнейший веб сервис, основанный на системе контроля версий Git. Позиционирует себя как огромную социальную сеть с открытым исходным кодом, созданную для совместной разработки проектов. GitHub предлагает все функции распределенного контроля версий (Git) и управления исходным кодом (SCM), а также добавляет свои собственные функции. Он обеспечивает контроль доступа и предоставляет несколько функции совместной работы, таких как отслеживание ошибок, запросы на создание нового функционала и управление задачами для каждого проекта. Сервис является бесплатным для публичных проектов, и насчитывает более 12 миллионов человек участвующих в 31 миллионе проектов. Веб интерфейс сервиса GitHub представлен на рисунке 1.3 [7].



Рисунок 1.3 – Веб интерфейс Git-репозитория

Bitbucket – распределенная система контроля версий, созданная для профессиональных команд. Она упрощает совместную работу команды над проектом благодаря гибким моделям развертывания. Единственное Git-решение, которое хорошо масштабируется и подходит для команд любых размеров. Также, как и GitHub, является бесплатной для публичных проектов с открытым исходным кодом. На рисунке 1.4 представлена панель инструментов, предлагаемых системой Bitbucket [8].



Рисунок 1.4 – Панель инструментов системы Bitbucket

Jira – это коммерческая, настраиваемая система отслеживания ошибок для управления проектами. Может использоваться для поддержки клиентов, поддерживает командную работу в режиме реального времени, предоставляет высокий уровень безопасности, API и множества плагинов. Поддерживает email-интеграцию и русский интерфейс. Рабочая доска системы Jira представлена на рисунке 1.5 [9].



Рисунок 1.5 – Рабочая доска системы Jira

GitLab – это веб-менеджер репозитория Git, предназначенный для написания кода, его развертывания и тестирования. Обеспечивает управление репозиторием с тонкоструктурным контролем доступа, обзорами кода, отслеживанием ошибок, лентами активности, вики-системами и непрерывной интеграцией. Разработан компанией GitLab Inc. и использует лицензию с открытым исходным кодом. Программное обеспечение написано на динамическом языке программирования Ruby и многопоточном языке Go. Используется такими организациями как IBM, Sony, NASA, Alibaba, Invincea, O’Reilly Media и CERN. Рабочая доска GitLab репозитория представлена на рисунке 1.6 [10].



Рисунок 1.6 – Рабочая доска GitLab

Все вышеперечисленные системы управления проектами базируются на использовании репозитория файлов Git.

Git – это система контроля версий (VCS) для отслеживания изменений в компьютерных файлах и координации работы с этими файлами среди множества людей. В основном используется для разработки программного обеспечения, но, также может использоваться для отслеживания изменений в любом другом наборе файлов. Будучи распределенной системой контроля версий, Git нацелен на скорость, целостность данных и поддержку распределенных нелинейных рабочих процессов. Модель данных, используемая Git, обеспечивает криптографическую целостность каждого бита вашего проекта. Каждый файл и его изменение проверяются и извлекаются контрольной суммой при необходимости возврата к предыдущей версии. Git был создан Линусом Торвальдсом в 2005 году для разработки ядра Linux. Как и ядро Linux, Git является бесплатным программным обеспечением, распространяемым в соответствии с GNU GPL (General Public License version 2) – открытое лицензионное соглашение GNU [11].

* 1. **SQL и NoSQL решения**

SQL или NoSQL база данных – это организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организованна таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части [12].

SQL БД или реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов – атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов). Строки таблицы называются записями, все записи таблицы имеют одинаковую структуру. Для идентификации записей используется первичный ключ. Первичным ключом называется набор полей таблицы, комбинация значений которых однозначно определяет каждую запись в таблице.

Не реляционная или NoSQL БД убирает все ограничения реляционной модели (недостаточная производительность, трудоемкое горизонтальное масштабирование, недостаточная производительность в кластере) и облегчает средства хранения и доступа к данным. Такие базы данных используют неструктурированный или безсхемный подход (создание структуры на лету), тем самым снимая ограничения жестких связей и предлагая различные типы доступа к специфическим данным. NoSQL базы данных не работают с реляционными моделями, используют много различных решений, каждое из которых работает немного по своему и служит специфической цели. Эти безсхемные решения снимают ограничения с формирования сущностей. NoSQL БД не используют общий формат запроса, такой как SQL в реляционных базах данных. Каждое NoSQL решение использует собственную систему запросов.

Приведем некоторые ключевые концепции реляционных и не реляционных баз данных. На рисунке 1.7 показана база данных, содержащая сведения о взаимоотношениях людей. Первый вариант – это безсхемная структура, построенная в виде графа, характерная для NoSQL-решений, второй вариант – представление тех же данных в структурированном виде, типичном для SQL.

Безсхемность означает, что двум документам в структуре данных NoSQL не обязательно иметь одинаковые поля, и они могут хранить данные разных типов. Например, массив объектов, набор полей которых не совпадает:

|  |
| --- |
| “var cars = [  {Model: “BMW”, Color: “Red”, Manufactured: 2016}, {Model: “Mercedes”, Type: “Coupe”, Color: “Black”, Manufactured: “1-2-2017”}];” |

При реляционном подходе, данные надо хранить в заранее спроектированной структуре, из которой эти данные потом можно будет извлечь [12].

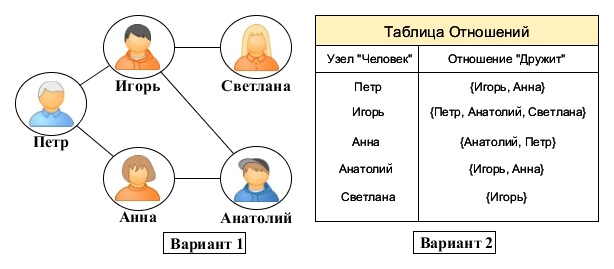


Рисунок 1.7 – Два варианта структур представления данных

Категория баз данных NoSQL заметно отличается от SQL БД. Базы данных NoSQL часто используется для описания систем управления данными, которые не относятся к SQL, или подхода к управлению данными, который предусматривает использование не только SQL. Существует ряд технологий категории NoSQL, включая базы данных документов, хранилища пар “ключ – значение”, хранилища семейств-столбцов, а также графовые базы данных, которые часто используются в играх, приложениях для работы с социальными сетями и приложениях управленческого характера.

В таблице 1 приведены основные различия между SQL и NoSQL базами данных [13].

Таблица 1 – Основные различия SQL и NoSQL

|  |  |  |
| --- | --- | --- |
| Характеристики | NoSQL | SQL |
| Тип транзакции | Поддержка ACID зависит от решения | Поддерживает транзакции ACID |
| Тип схемы СУБД | Динамические или гибкие схемы | Строгие, структурированные схемы |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Характеристики | NoSQL | SQL |
| Тип схемы СУБД | База данных является схемой-агностиком, и схема диктуется приложением. Это обеспечивает гибкость и высокую итеративную разработку | Схема должна поддерживаться и хранится в синхронизации между приложением и базой данных |
| Модель (структура) данных | Хранит данные в документах JSON, парах ключ-значение, хранилищах семейств-столбцов и графах | Хранит данные в таблице |
| Свойства данных | Предлагает гибкость, поскольку не каждой записи нужно хранить те же свойства | Отлично подходит для решений, где каждая запись имеет одинаковые свойства |
| Новые свойства могут быть добавлены на лету, без изменения схем или рабочих запросов | Добавление нового свойства может потребовать изменение схем или данных обратной засылки, поскольку не поддерживается гибкая структура данных |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Характеристики | NoSQL | SQL |
| Свойства данных | Отношения часто захватываются путем денормализации данных и представления всех данных для объекта в одной записи | Отношения часто захватываются в нормализованной модели с использованием объединений для разрешения ссылок между таблицами |
| Хорошо подходит для полу структурированных, сложных или вложенных данных | Хорошо подходит для структурированных данных |
| Производительность | Производительность может быть увеличена за счет уменьшения согласованности, если это необходимо | Производительность зависит от того, насколько быстро выполняется запись, так как обеспечивается сильная согласованность. Производительность может быть увеличена с помощью масштабирования доступных ресурсов |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Производительность | Информация об объекте обычно находится в одной записи, поэтому обновление может произойти за одну операцию | Информация об объекте может быть распределена по многим таблицам, это требует многих соединений для завершения запроса |
| Последовательность и доступность | Возможна поддержка сильной согласованности | Обеспечение жесткой последовательности |
| Согласованность, доступность и производительность могут предоставляться в соответствии с требованиями приложения | Приоритетность согласованности над доступностью и производительностью |

В зависимости от того что необходимо разрабатывать либо исходя из требований к приложению или системе, можно выбрать одно из имеющихся решений. В некоторых случаях применяются оба подхода для еще более гибкой и производительной работы.

* 1. **Графовые СУБД**

Графовые базы данных – NoSQL решение, разновидность баз данных с реализацией сетевой модели в виде графа и его обобщений с использованием графовой СУБД, которая дает новые возможности для работы со связанными данными. Графовую модель данных обычно рассматривают как обобщение RDF – модели или сетевой модели данных [14].

Система управления графовыми базами данных (графовые базы данных) поддерживает методы создания, чтения, изменения, и удаления, основанные на графовой модели данных. Графовые базы данных, как правило, поддерживают систему транзакций реального времени (OLTP). Соответственно, они оптимизированы для выполнения транзакций и спроектированы с учетом транзакционной целостности и оперативности.

Имеются две особенности графовых баз данных, которые необходимо учитывать при рассмотрении применяемой ими технологии:

1. принцип хранения. Некоторые графовые базы данных используют специализированные хранилища графов, предназначенные и оптимизированные для хранения и обработки именно графов. Но такую технологию хранения используют не все графовые базы данных. Некоторые сериализуют графы и размещают их в реляционной, объектно-ориентированной или какой-то другой базе данных или хранилище;
2. порядок обработки. Некоторые определения требуют, чтобы графовая база данных использовала смежность без индексов, то есть физическое соединение друг с другом.

Взаимосвязи в графовой модели данных являются гражданами первого сорта. Здесь к ним относятся не так, как в других системах управления базами данных, где для отображения взаимосвязей применяются такие механизмы, как внешние ключи или внешние операции, например, MapReduce. Собирая абстракции узлов и взаимосвязей в связанные структуры, графовая база данных позволяет строить модели любой сложности, лучше всего отражающие предметную область. Полученные модели проще и в то же время нагляднее, чем те, что создаются с помощью традиционных реляционных баз данных или других NOSQL-хранилищ [15].

В таблице 2 приведены некоторые характеристики и поддерживаемые технологий графовых СУБД, таких как Neo4j, InfiniteGraph и AllegroGraph.

Таблица 2 – Сравнительные характеристики графовых СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | InfiniteGraph | Neo4j | AllegroGraph |
| Источник | Собственная разработка с открытым исходным кодом | Открытый исходный код | Собственная разработка |
| Граф модель | Ориентированный граф | Ориентированный граф | RDF, XML |
| Модель схемы | Гибридная | Собственная модель | - |
| API | Java, Gremlin, Blueprints | Java, Gremlin, Blueprints | Java, Python, Ruby, C#, Perl, Clojure, Lisp |
| Язык запросов | Gremlin, Blueprints | Cypher | SPARQL, Prolog, RDFS++ |
| Поддерживаемые платформы | Windows, Linux, Mac OS X | Windows, Linux, Mac OS X | Windows, Linux, Mac OS X, Free BSD, Solaris |
| Транзакции | ACID | Поддержка ACID | ACID |
| Инструменты визуализации | JSON, GraphML, IG Visualizer | Инструменты веб администрирования | - |
| Резервное копирование / Восстановление | Инкрементальное резервное копирование, полное восстановление | Резервное копирование и восстановление поддерживается только в версиях для предприятий | Полное резервное копирование и восстановление |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | InfiniteGraph | Neo4j | AllegroGraph |
| Язык реализации | Java, C++, Groovy, Scala | Java, Scala | Java, Python, Ruby, Perl, C#, Clojure |

* + 1. **Механизмы вычисления графов**

Такие механизмы позволяют выполнять глобальные графовые вычислительные алгоритмы для больших наборов данных. Они предназначены для решения таких задач, как идентификация кластеров данных или получение ответов на такие вопросы, как: “Сколько всего взаимосвязей, сколько их в среднем, полна ли социальная сеть?”

Из-за своей направленности на глобальные запросы механизмы вычисления графов, как правило, оптимизированы для сканирования и пакетной обработки больших объёмов информации, и в этом отношении они похожи на другие технологии пакетного анализа, такие как интеллектуальный анализ данных или аналитическая обработка в реальном времени (OLAP), используемые в реляционном мире [17].

Наиболее известными механизмами вычисления графов являются такие механизмы как:

* Cassovary;
* Pegasus;
* Apache Giraph.

Cassovary – простая библиотека обработки больших графов, разработанная для виртуальной машины Java (JVM). Большинство графовых JVM библиотек являются гибкими, но не эффективными с точки зрения используемого пространства. Так как Cassovary разработана с нуля, она имеет возможность эффективно обрабатывать миллиарды узлов и связей графа. Типичными примерами использования такой библиотеки являются крупномасштабные построения графов и анализ большой сети. Cassavary написана на мультипарадигмальном языке программирования Scala и может использоваться с любыми JVM – поддерживающими языками. Библиотека поставляется с некоторыми общими структурами данных и алгоритмами [18].

Pegasus – это система для масштабирования графов, написанная полностью на объектно-ориентированном языке программирования Java. Система работает параллельно-распределенным образом поверх Hadoop. Hadoop – это платформа облачных вычислений, а также реализация открытого исходного кода модели распределенных вычислений MapReduce, которая изначально была разработана корпорацией Google для обработки данных в режиме реально времени. Для важных графовых интеллектуальных задач, Pegasus предоставляет крупномасштабные алгоритмы, такие как:

* Degree;
* PageRank;
* RWR (Random Walk with Restart);
* Radius и другие.

Pegasus обладает такими преимуществами как: пакетом для крупномасштабного графопостроения, параллельными алгоритмами на Hadoop и открытым исходным кодом [19].

Apache Giraph – это итеративная система обработки графов в режиме реального времени, созданная для высокой масштабируемости и анализа данных социальных сетей.

В настоящее время, она используется компаниями Yahoo, Facebook и Twitter для анализа социальных графов, сформированных пользователями и их соединениями. Giraph возникла как аналог системы с открытым исходным кодом Pregel, архитектура обработки графов которой была разработана корпорацией Google [20].

Некоторые механизмы вычисления включают в себя и средства хранения графов, а другие (большинство) заботятся только об обработке данных, получаемых из внешнего источника, а затем возвращают результаты для сохранения в другом месте. Рисунок 1.8 демонстрирует типовую архитектуру развертывания механизмов вычисления графов.



Рисунок 1.8 – Архитектура развертывания механизмов вычисления графов

Схема включает в себя систему записи (SOR) базы данных со свойствами OLTP (к примеру, Oracle или Neo4j), которая обслуживает запросы и отвечает на запросы, поступающие от приложения и, в конечном счете, от пользователей. Задания на извлечения, преобразования и загрузку данных перемещают данные из системы записи базы данных, в механизм вычисления графов для выполнения анализа и автономных запросов [21].

* + 1. **Преимущества графовых баз данных**

Практически любою деятельность можно представить в виде графа. Предоставляемый графовыми базами данных новый способ моделирования данных сам по себе не дает достаточного основания для замены давно устоявшихся и понятных платформ обработки данных. Этот способ должен давать незамедлительную и очень значительную практическую пользу. Мотивация перехода на графовые базы данных заключается в том, что при определенной модели данных такой переход будет давать существенное увеличение производительности на один и более порядков.

Ощутимый прирост производительности при использовании графовых баз данных достигается, если работа ведется со связанными данными, по сравнению с теми же реляционными базами данных или NoSQL-хранилищами. В отличие от реляционных БД, где учет взаимосвязей большого объёма данных ощутимо ухудшает производительность запросов, производительность графовых БД при росте объёма данных остается неизменной. Это связано с тем фактом, что запросы в графовой БД локализуются в определенной части графа. В итоге, время выполнения каждого запроса зависит только от размера части графа, в которой происходит поиск, а не от его общего размера.

Помимо роста в производительности, графовые базы данных предоставляют гибкую модель данных и способ развертывания, который соответствует современным способам развертывания ПО. Структура данных должна соответствовать изменяющимся потребностям, а не навязываться заранее и оставаться неизменной. В графовых БД эта задача легко решается, графовая модель данных учитывает потребности бизнеса, что и дает ей возможность изменяться со скоростью изменения самого бизнеса.

Возможность расширения, означает что можно добавлять и дополнять новые виды взаимосвязей, новые узлы, новые метки, а также новые подграфы в существующую структуру и что самое важное – это происходит без нарушения существующих запросов и функционала приложения. Такая возможность положительно влияет на производительность процесса разработки и снижает риски для проекта. Благодаря гибкости графовой БД, нет необходимости заранее моделировать задачу в мельчайших подробностях, что очень неудобно, поскольку в бизнесе требования очень часто меняются. Способность графов к расширению также позволяет уменьшить количество миграций, это снижает нагрузку при обслуживании данных и уменьшает риск потерь [15, 16].

* + 1. **Графовая СУБД Neo4j**

Ввиду исходных данных, для разработки графовой базы данных системы управления проектами будет использоваться графовая СУБД Neo4j.

СУБД Neo4j – это бесплатная база данных NoSQL с открытым исходным кодом, которая реализована на языках Java и Scala. СУБД Neo4j используется сегодня сотнями тысяч организаций, практический во всех отраслях, таких как аналитика программного обеспечения, научные исследования, маршрутизация, организационное и проектное управление, социальные сети и во многих других.

В графовой базе данных Neo4j есть одно основное правило: “Нет потерянных ссылок”. Поскольку отношение всегда имеет начальный и конечный узлы, нельзя удалить узел, не удалив связанные с ним отношения. Также существующее отношение никогда не укажет на несуществующую конечную точку (узел). На рисунке 1.9 изображен пример графовой модели данных Neo4j.

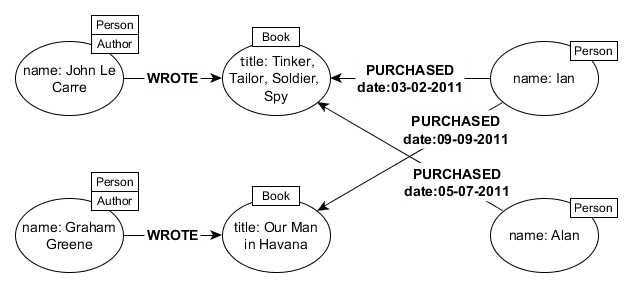


Рисунок 1.9 – Пример графовой модели данных Neo4j

СУБД Neo4j эффективно реализует графовую модель вплоть до уровня хранилища. В отличие от обработки графов или библиотек в памяти, Neo4j обладает полными характеристиками базы данных, включая соответствие транзакциям ACID, поддержку кластера и переключение при выполнении во время выполнения, что позволяет использовать данные графа в производственных сценариях. Встроенный механизм хранения и обработки графовых данных Neo4j обеспечивает постоянную производительность в режиме реального времени, помогая предприятиям создавать интеллектуальные приложения для решения современных проблем с данными. Графовая СУБД Neo4j использует собственное граф-хранилище, специально предназначенное для хранения и управления взаимосвязанными данными. Каждая часть данных имеет явную связь со всеми связанными объектами, то есть запросы базы данных могут игнорировать все, что не связано, а не обходить весь набор данных. Результатами такого хранилища являются – высокая скорость и хорошая масштабируемость. Neo4j предоставляет гибкую, “удобную для пользователя” модель данных, в которой типы данных или их свойства могут быть добавлены или изменены в любое время, что приводит к значительному сокращению времени разработки и подлинной гибкой итерации.

Некоторые особенности (рис.1.10) делают Neo4j очень популярной среди пользователей, разработчиков и администраторов баз данных:

* материализация отношений во время создания, не приводящая к увеличению времени выполнения сложного запроса;
* константное время обхода связей графа, как в глубину, так и в ширину благодаря эффективному представлению узлов и отношений;
* все отношения в Neo4j одинаково важны и быстры, что позволяет впоследствии материализовать и использовать новые отношения для “сокращения” пути обхода графа и ускорить получение данных домена при возникновении новых потребностей;
* компактное хранение и кэширование, которое приводит к эффективному масштабированию и миллиардам узлов в одной базе данных на среднем оборудовании;
* высокая скорость загрузки любого объёма данных не требующая большой вычислительной мощности ЭВМ;
* простота в использовании благодаря языку графовых запросов Cypher.

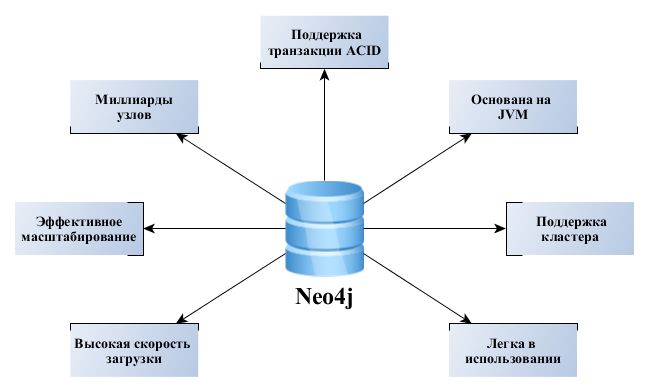


Рисунок 1.10 – Особенности Neo4j

Помимо этого, в графовой СУБД Neo4j присутствует реализация графического интерфейса. Он дает возможность удобно просматривать графовые данные, которые хранятся в базе. Пример графического интерфейса для изображения данных в Neo4j представлен на рисунке 1.11.



Рисунок 1.11 – Графический интерфейс изображения данных в Neo4j

Neo4j распространяется в двух версиях: версия для свободного пользования – высокопроизводительная, полностью ACID-транзакционная база данных и корпоративная версия – включает в себя (но не ограничивается) всю функциональность версии для свободного пользования в дополнение к масштабируемой кластеризации, отказоустойчивости, высокой готовности, оперативной архивации и всестороннему мониторингу [22].

1. **ИСПЛЬЗУЕМЫЕ ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ**
   1. **Сравнение MySQL и Neo4j**

В качестве реляционной БД, для сравнения с Neo4j, была выбрана база данных MySQL.

MySQL – одна из самых популярных реляционных СУБД корпорации Oracle с открытым исходным кодом, основанная на структурированном языке запросов (SQL). БД MySQL хранит данные в виде связанных таблиц и обычно используется для разработки веб-приложений. Предоставляет API для таких языков как C, C++, Eiffel, Java, Perl, PHP и Python. MySQL поддерживает работу в таких операционных системах как UNIX, Windows и Mac OS. Обеспечивает высокую производительность при работе со структурированными данными [23].

Что бы сравнить производительность баз данных необходимо заполнить их одними и теми же данными, при этом объём данных должен быть существенный, так как при малом объеме разница будет неощутима. Исходя из требований, была выбрана тестовая предметная область – социальная сеть и составлена ER – диаграмма (рис.2.1), на основе которой были созданы реляционная и графовая модели.

Обе БД были заполнены следующим количеством данных:

* 100 000 пользователей;
* 200 000 групп;
* 300 000 фотографий;
* 250 000 аудиозаписей;
* 1 000 000 друзей;
* 4 000 000 сообщений;
* 350 000 аудиозаписей пользователей;
* 400 000 групп пользователей;
* 400 000 фотографий пользователей.

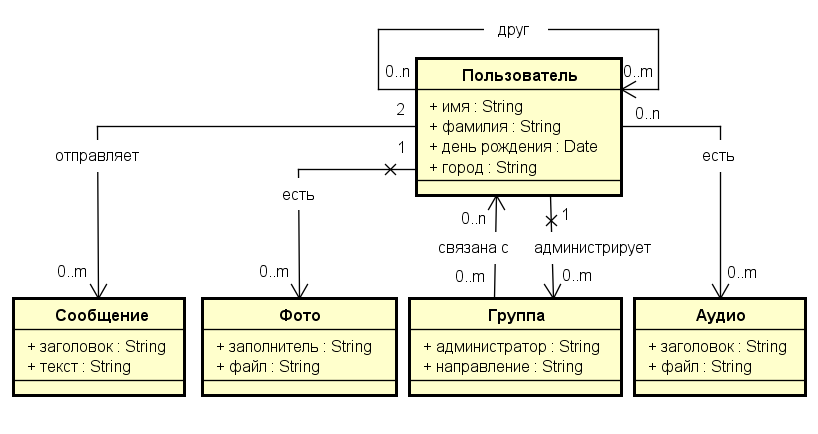


Рисунок 2.1 – ER – диаграмма реляционной и графовой модели

Размер базы данных MySQL составил 351.5 мегабайт, а размер БД Neo4j – 3.45 гигабайт. Разница в объеме между базами данных оказалась довольно ощутимой, порядка трех гигабайт. Это связано с тем, что было использовано много полей с текстовой информацией.

Эксперимент проводился на ЭВМ со следующими конфигурациями:

* операционная система – Windows 8.1;
* тип системы – 64-разрядная;
* процессор – Intel ® Core ™ i5 – 3230M CPU @ 2.60 GHz;
* оперативная память (ОЗУ) – 6.00 Гигабайт;
* жесткий диск – HGST SAS 3.0 900 Гигабайт.

Эксперимент заключался в измерении времени поиска общего количества фотографии у пользователей, которые администрируют хотя бы одну группу, в зависимости от диапазона значений идентификаторов пользователя. Поскольку такой поиск осуществляется по общим взаимосвязям, он является сложным и отражает более наглядное время поиска для сравнения.

Для того что бы результаты поиска были максимально точными, эксперимент проводился при условии, что базы данных MySQL и Neo4j не работали одновременно на одной ЭВМ.

Результаты проведенного эксперимента отражены в таблице 3.

Таблица 3 – Результаты измерения времени поиска

|  |  |  |
| --- | --- | --- |
| Диапазон идентификатора < | Время для MySQL, с | Время для Neo4j, с |
| 10 | 0,12 | 10,767 |
| 100 | 0,69 | 10,706 |
| 1 000 | 5,879 | 10,884 |
| 5 000 | 24,668 | 12,245 |
| 10 000 | 52,462 | 12,280 |
| 30 000 | 154,534 | 13,352 |
| 60 000 | 296,369 | 14,545 |
| 90 000 | 489,830 | 18,058 |

По результатам таблицы 3 была построена диаграмма зависимостей (рис. 2.2) для проведенного эксперимента.

Рисунок 2.2 – Диаграмма измерения времени поиска

Проведенный эксперимент показал, что графовая база данных производительнее по времени поиска при большом объёме данных, но при этом занимает приличное место на жестком диске (3.45 гигабайт) в отличие от реляционной БД (351.5 мегабайт). Также можно сказать, что для систем с небольшим объёмом данных лучше использовать реляционную базу данных, так как при таких условиях она будет справляться лучше. Эксперимент и его результаты были взяты из источника [24].

* 1. **Язык запросов Cypher**

Cypher – это декларативный графовый язык запросов, который позволяет выразительно и эффективно строить запросы и обновлять графовые хранилища. Относительно простой, но все же очень мощный язык. Очень сложные запросы к базам данных могут быть легко выражены через Cypher. Его конструкции основаны на английской прозе и аккуратной иконографии, которая помогает сделать запросы более понятными.

Будучи декларативным языком, Cypher сосредоточен на ясности выражения того, что извлекать из графа, а не та том, как это извлекать. Он отличается от императивных языков, таких как Java, скриптовых языков, таких как Gremlin, и привязок JRuby. Такой подход делает оптимизацию запросов деталями реализации, а не обременяет пользователя ею и требует от нее обновления всех обходов только потому, что изменилась физическая структура базы данных (новые индексы и т. д.).

Cypher вдохновлен рядом различных подходов и опирается на сложившиеся практики выразительного запроса. Большинство ключевых слов, таких как WHERE и ORDER BY, заимствованы из SQL. Совмещение с образцом заимствует подходы к выражению из SPARQL. Некоторые из выражений были заимствованы из таких языков, как Haskell и Python [25].

Cypher предоставляет пользователю (или приложению, действующему от имени пользователя) возможность задавать шаблон для поиска данных. Проще говоря, можно попросить базу данных “найти что-то похожее на это”.

Для описания того, “как это должно выглядеть” используется ASCII-графика. На рисунке 2.3 изображен шаблон, который описывает трех друзей.

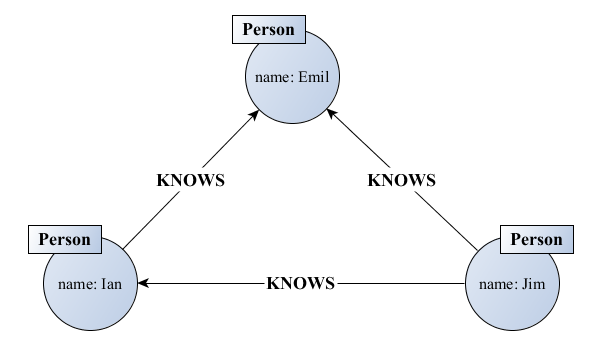


Рисунок 2.3 – Модель графа, изображенная схемой

А вот его эквивалент на языке графовых запросов Cypher:

|  |
| --- |
| (emil: Person {name: ‘Emil’})  <-[:KNOWS]–(jim: Person {name: ‘Jim’})  –[:KNOWS]->(ian: Person {name: ‘Ian’})  –[:KNOWS]->(emil) |

Этот шаблон описывает маршрут, соединяющий узел с именем Jim с двумя другими узлами Ian и Emil, которые также связаны соединением, идущим от узла Ian к узлу Emil. «ian», «jim» и «emil» являются идентификаторами. Идентификаторы позволяют ссылаться на узлы в описании модели и обойти тот факт, что язык запросов имеет только одно измерение (его текст записывается слева направо), в то время как схема графа использует два измерения.

Шаблоны графов, представленные ASCII-графикой, являются основой Cypher. Запросы на Cypher прикрепляют одну или несколько частей шаблона к определенным местам графа с помощью предикатов, а затем перемещают незафиксированные части, пытаясь найти соответствие.

Как и большинство языков запросов, Cypher состоит из фраз. В таблице 4 приведено описание фраз языка Cypher, наиболее часто используемых в запросах.

Таблица 4 – Фразы языка запросов Cypher

|  |  |
| --- | --- |
| Фраза | Описание |
| MATCH | Основа большинства Cypher запросов. Шаблон для сопоставления и самый распространённый способ для получения данных из графовой структуры |
| RETURN | Определяет, какие узлы, взаимосвязи и свойства в совпавших данных должны быть возвращены пользователю в результате запроса |
| WHERE | Определяет критерии совпадения результатов для фильтрации извлекаемой информации |
| CREATE и CREATE UNIQUE | Создает узлы, метки, взаимосвязи и их свойства |
| MERGE | Гарантирует, что заданный шаблон будет существовать в графе либо за счет использования уже найденных в графе узлов и взаимосвязей, соответствующих заданным свойствам, либо за счет создания новых узлов и взаимосвязей |
| FOREACH | Вносит изменения в каждый из элементов списка |
| DELETE | Удаляет узлы, метки, взаимосвязи и их свойства |
| SET | Устанавливает значение свойств |
| WITH | Объединяет части запроса в цепочку и передает результаты от одной части запросы к другой. Работает подобно именованным каналам в Unix |

Продолжение таблицы 4

|  |  |
| --- | --- |
| Фраза | Описание |
| START | Явно указывает одну или несколько отправных точек, узлов или взаимосвязей в графе (Фраза START признана устаревшей, и вместо нее рекомендуется явно указывать отправные точки во фразе MATCH). |
| LIMIT | Ограничивает количество строк при выводе результата запроса |
| UNION | Объединяет результаты двух или более запросов |
| ORDER BY | Фраза, которая следует после RETURN или WITH. Сортирует результат запроса определенным образом |
| REMOVE | Удаляет свойства и метки из узлов и отношений |
| SKIP | Фраза, определяющая с какой строки нужно начать обрабатывать результат запроса, включая и строки на выходе |
| UNWIND | Развертывает список в последовательность строк |
| LOAD CSV | Используется при импортировании данных из CSV-файлов |
| DETACH DELETE | Удаляет узел или набор узлов с автоматическим удалением всех связанных с этим узлом отношений |
| CALL…YIELD | Вызывает процедуру, развернутую в базе данных |

Пример использования нескольких фраз в одном запросе языка Cypher:

|  |
| --- |
| MATCH (a: Person)–[:KNOWS]->(b)–[:KNOWS]->(c), (a)–[:KNOWS]->(c)  WHERE a.name = ‘Jim’  RETURN b, c  LIMIT 25 |

В качестве результата, такой запрос вернет нам прямых друзей пользователя Jim в пределах 25-ти строк [15].

Объединяя и комбинируя фразы, описанные в таблице 4, можно составлять более сложные и специфические запросы для извлечения необходимых данных.

Помимо фраз, Cypher поддерживает использование операторов и функций. Операторы языка Cypher и их описание представлены в таблице 5.

Таблица 5 – Операторы языка Cypher

|  |  |  |
| --- | --- | --- |
| Оператор | Тип оператора | Описание |
| % | Математический | Деление по модулю |
| \* | Математический | Умножение |
| + | Математический | Сложение |
| + | Строковый | Конкатенация |
| + | Список | Конкатенация списков |
| - | Математический | Вычитание или унарный минус |
| . | Общий | Доступ к свойствам |
| / | Математический | Деление |
| < | Сравнительный | Меньше чем |
| <= | Сравнительный | Меньше чем или равно |
| <> | Сравнительный | Неравенство |
| = | Сравнительный | Равенство |
| > | Сравнительный | Больше чем |
| ^ | Математический | Возведение в степень |
| >= | Сравнительный | Больше чем или равно |
| AND | Логический | Конъюнкция |
| CONTAINTS | Строковое сравнение | Поиск с учетом регистра |
| DISTINCT | Общий | Удаление дубликатов |
| ENDS WITH | Строковое сравнение | Поиск суффикса с учетом регистра |
| IN | Список | Проверка существования элемента списка |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| Оператор | Тип оператора | Описание |
| IS NOT NULL | Сравнительный | Ненулевая проверка |
| IS NULL | Сравнительный | Проверка на ноль |
| NOT | Логический | Отрицание |
| OR | Логический | Дизъюнкция |
| STARTS WITH | Строковое сравнение | Поиск префикса с учетом регистра |
| XOR | Логический | Исключающее или |
| [] | Общий | Подстрочный индекс, доступ к динамическим свойствам |
| [] | Список | Подстрочный индекс, доступ к элементу (элементам) в списке |

Функции языка Cypher делятся на следующие типы:

* предикатные;
* скалярные;
* агрегирующие;
* функции списка;
* математические;
* строковые;
* пространственные;
* пользовательские.

Пользовательские функции пишутся на языке Java, развертываются в базе данных и вызываются также, как и любые другие функции Cypher. В качестве примера, в таблице 6 приведен список некоторых функции языка Cypher с их описанием.

Таблица 6 – Некоторые функции языка Cypher

|  |  |  |
| --- | --- | --- |
| Функция | Тип функции | Описание |
| Abs() | Математический | Возвращает абсолютное значение числа |
| Acos() | Математический | Возвращает арккосинус выражения |
| All() | Предикатный | Проверяет, сохраняется ли предикат для всех элементов в списке |
| Collect() | Агрегирующий | Возвращает список, содержащий все собранные значения |
| Distance() | Пространственный | Возвращает число с плавающей точкой, представляющее геодезическое расстояние между любыми двумя заданными точками |
| Extract() | Список | Возвращает одно свойство или значение функции из списка узлов или отношений |
| Exists() | Предикатный | Возвращает истину (true), если совпадение шаблона существует в графе, или свойство существует в узле, отношении или карте |
| Filter() | Список | Возвращает все элементы в списке, соответствующие предикату |
| Id() | Скалярный | Возвращает идентификатор отношения или узла |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| Функция | Тип функции | Описание |
| Coalesce() | Скалярный | Возвращает первое ненулевое значение в списке переданных ей выражений |
| Length() | Скалярный | Возвращает длину пути |
| Substring() | Строковый | Возвращает подстроку заданной строки, начиная с длины нулевого индекса |

Пример вызова одной из функций Cypher выглядит следующим образом:

|  |
| --- |
| MATCH (n:Person)  RETURN Collect(n.age) |

Результатом вызова функции Collect() будет список, содержащий все собранные значения по свойству “age”.

Полный список всех функции которые поддерживает язык Cypher находится в источнике [25].

1. **РЕАЛИЗАЦИЯ**

В данном разделе описаны действия, связанные с разработкой графовой базы данных на СУБД Neo4j для системы управления проектами, скриптов для заполнения графовой БД и создание запросов на языке Cypher.

* 1. **Разработка графовой базы данных на СУБД Neo4j**

Исходя из разделов 1.1, 1.2, и 1.3, графовая база данных для системы управления проектами должна упрощать управление, контроль и отслеживание изменений в имеющихся или вновь созданных проектах. Учитывая эти особенности, разрабатываемая графовая БД должна хранить следующие данные:

* наименования и описания проектов, их заказчиков и руководителей;
* наименования и описания задач, подзадач проекта;
* исполнителей (разработчиков) назначенных на задачи;
* совершённые изменения и их описание на этапе выполнения задач;
* URL адреса репозиториев в которых были созданы задачи или совершены их изменения на этапе выполнения.

Как упоминалось в разделе 1.5.3, графовая СУБД Neo4j хранит данные в гибком виде – “удобном для пользователя”, используя для этого вершины и связи, поэтому СУБД Neo4j не нуждается в проектировании логической схемы модели данных, как это бы потребовалось при реляционном подходе. Это приводит к упрощенной разработке базы данных. Необходимо лишь создать и наименовать вершины и связи с их метками, а также при надобности добавить им свойства. Соединить вершины между собой созданными связями. Пройдя описанные этапы, разработанная модель будет готова для дальнейшего использования и извлечения необходимых данных из базы. При необходимости, разработка может состоять из нескольких и более итерации для дополнения или модификации модели.

Схема разработанной базы данных для системы управления проектами представлена на рисунке 3.1.

|  |
| --- |
| Рисунок 3.1 – Схема графовой модели БД для системы управления проектами |



Схема состоит из 13 вершин и 23 связей. За каждой вершиной закреплена метка, и каждая связь имеет свое свойство и направление.

Вершины графовой базы данных хранят такие данные как:

* ФИО заказчиков (Customer);
* наименования проектов (Project);
* статусы проектов (Project\_status);
* типы проектов (Project\_type);
* коды проектов (Project\_code);
* ФИО руководителей проектов (Master);
* наименования задач (Issue);
* типы задач (Issue\_type);
* приоритеты задач (Issue\_priority);
* статусы задач (Issue\_status);
* ФИО разработчиков (Developer);
* совершённые изменения и их описание (Commit);
* наименования и URL адреса репозиториев (Repository).

Свойства связей между вершинами, обеспечивают общее понимание разработанной графовой схемы системы управления проектами. К примеру, вершины Master (руководитель проекта), Project (проект), Issue (задача) и Developer (разработчик) связаны между собой такими свойствами как: Creates (создает), Consists\_of (состоит из), Appoints (назначает) и Executes\_the (выполняет). Следуя схеме (рис.3.1), нетрудно понять, что руководитель создает проект и задачи, назначает разработчиков, которые будут выполнять конкретную задачу или задачи, а сам проект состоит из задач.

Разработанная таким образом графовая модель, гарантирует извлечение необходимых связанных данных при корректно построенных запросах, позволяет легко отслеживать внесенные изменения при выполнении задач проекта и контролировать состояние проектов в целом.

* 1. **Создание скриптов для заполнения графовой БД**

Для заполнения базы данных на СУБД Neo4j существует много способов, но все они, так или иначе, используют язык графовых запросов Cypher. В текущей работе, для заполнения БД данными, был использован стандартный способ, предоставляемый СУБД – браузер Neo4j с его рабочей станцией.

Браузер Neo4j становиться доступным после запуска СУБД и представляет собой веб-страницу с адресом: <http://localhost:7474/browser/>. Адрес браузера не нужно запоминать, достаточно просто кликнуть на него, как только Neo4j запуститься (рис.3.2).



Рисунок 3.2 – Стартовое окно СУБД Neo4j

Перейдя по указанному адресу, мы попадаем на нашу рабочую станцию (рис.3.3).

Рабочая станция располагает всем необходимым для работы с базой данных. Есть возможность посмотреть информацию о данных, которые хранятся в БД, а также системную информацию, такую как:

* детальный размер хранилища;
* распределенные ID;
* транзакции;
* статистику кэша страницы.



Рисунок 3.3 – Рабочая станция браузера Neo4j

Помимо этого, можно воспользоваться примерами скриптов, предоставляемыми Neo4j, импортировать собственные скрипты, настраивать графический интерфейс пользователя, просматривать документацию Neo4j, синхронизироваться с облачными сервисами и управлять локальными данными. Для ввода запросов на языке Cypher и других команд, связанных с управлением рабочей станции, имеется специальная строка-терминал, а для вывода результатов этих запросов/команд, как упоминалось в разделе 1.5.3, реализован графический интерфейс изображения данных.

Что бы создать скрипты, была использована одна из фраз языка Cypher – CREATE. Фраза CREATE, описанная в разделе 2.2, создает граф элементы, такие как узлы, метки, взаимосвязи и их свойства.

Синтаксическая структура создания узла следующая:

|  |
| --- |
| CREATE  (<идентификатор>: <имя\_метки>  {<имя\_свойства\_1>: ‘<значение\_свойства\_1>’,  …  ,<имя\_свойства\_N>: ‘<значение\_свойства\_N>’}) |

Идентификаторы, имена меток, свойств и их значений можно задать как в текстовом, так и в числовом формате.

Создание узлов Проект (Project) и Заказчик (Customer) выглядит следующим образом:

|  |
| --- |
| CREATE  (p1:Project{name: ‘Audio portal’}),  (p2:Customer{firstname: ‘Ivan’, lastname: ‘Bulov’}) |

Для того что бы создать отношение между двумя узлами, используется следующий синтаксис:

|  |
| --- |
| CREATE  (<идентификатор\_узла\_A>)-[<идентификатор\_отношения>:<имя\_метки\_отношения>]->  (<идентификатор\_узла\_Б>) |

Идентификатор отношения и имя метки отношения также можно задать в числовом или текстовом формате. Идентификатор отношения не является обязательным, при создании отношения его можно опустить, в этом случае Cypher сам задаст отношению уникальный идентификатор по умолчанию. Символы “-” и “->” указывают направление отношения. Необходимо понимать, что нельзя создать отношение между несуществующими узлами и учитывать это при написании скрипта.

Создание отношения “Заказывает (Orders)” между узлами Проект (Project) и Заказчик (Customer) выглядит так:

|  |
| --- |
| (p2)-[:Orders]->(p1) |

В данном случае p2 и p1 являются идентификаторами узлов Проект и Заказчик, отношение направлено от узла p2 к узлу p1, идентификатор отношения был опущен.

В результате было создано два скрипта для двух проектов: “DS\_IO\_API” и “AUDIO\_PORTAL”. Скрипты сохранены в файлах с расширением .txt и импортированы в базу данных с помощью рабочей станции браузера Neo4j. На рисунке 3.4 изображен результат выполнения скриптов. Код скриптов написан на языке Cypher и находится в приложениях A и B.



Рисунок 3.4 – Результат выполнения скриптов

* 1. **Создание запросов на языке Cypher**
     1. **Запрос 1. Получение сведений о проектах**

Исходя из разработанной схемы графовой БД для системы управления проектами (раздел 3.1), у проектов есть свои заказчики и руководители, они разделены на задачи, а также в БД хранятся сведения о кодах, статусах и типах проектов. Для получения такой информации, строится следующий запрос:

MATCH

(p:Project)--(n)

RETURN p AS Project, n AS Description

Такой несложный запрос вернет те вершины и отношения, которые связаны с вершиной Проект (Project). Результаты запроса в текстовом виде представлен на рисунке 3.5.



Рисунок 3.5 – Результат первого запроса в текстовом виде

Текстовый результат запроса отражает лишь наименование проектов (“DS\_IO\_API” и “AUDIO\_PORTAL”) и свойств всех вершин, связанных с найденными проектами в БД, но он не отражает отношений связанных вершин. Для более наглядного изображения результата запроса 1 согласно созданной модели, приведен результат запроса в графическом виде (рис. 3.6).



Рисунок 3.6 – Результат первого запроса в графическом виде

Графический вид запроса 1 позволяет более наглядно отобразить полученный результат. Дает возможность увидеть сколько проектов храниться в БД на текущий момент, оценить количество связанных с проектами вершин, а также отразить их отношения.

* + 1. **Запрос 2. Получение описания задачи**

В разработанной графовой БД для системы управления проектами, задача характеризуются своим статусом, приоритетом, типом, разработчиком, выполняющим задачу и репозиторием где задача была создана. Для получения такой информации, строится следующий запрос:

|  |
| --- |
| MATCH  (i:Issue)-[:Type\_of\_issue]-(t:Issue\_type),  (i:Issue)-[:Status\_of\_issue]-(s:Issue\_status),  (i:Issue)-[:Priority]-(p:Issue\_priority),  (i:Issue)-[:Link\_to]-(r:Repository),  (i:Issue)-[:Executes\_the]-(d:Developer)  WHERE i.name = "Parsing parameters"  RETURN i AS issue, t AS type, s AS status, p AS priority,  d AS developer, r AS repo |

Результаты выполнения запроса 2 представлены на рисунках 3.7 и 3.8.



Рисунок 3.7 – Результат второго запроса в текстовом виде



Рисунок 3.8 – Результат второго запроса в графическом виде

* + 1. **Запрос 3. Получение истории изменения задачи**

Важной информацией при управлении проектами является история изменения задач, то есть, как проходит их выполнение, на какой стадии выполнения находятся назначенные разработчики и вся остальная информация, связанная с решением задач.

Для получения истории изменений конкретной задачи, имени разработчика, а также репозитория где совершались изменения, строится следующий запрос:

|  |
| --- |
| MATCH  (i:Issue)-[:Start\_of]-(c:Commit),  (d:Developer)-[:Executes\_the]-(i:Issue),  (c:Commit)-[:Link\_to]->(r:Repository),  (c:Commit)-[:Next\*]->(n)  WHERE i.name = "SEEK function for DA module"  RETURN d AS Developer, i AS Issue, c AS Commit, r AS Repo, n AS Next |

В данном запросе используется конструкция [:Next\*], она позволяет не беспокоиться о длине отношения “Next” и делает эту длину переменной. Это необходимо из-за того, что у разных задач может быть разная длина истории изменений. Результаты выполнения запроса представлены на рисунках 3.9 и 3.10.



Рисунок 3.9 – Результат третьего запроса в текстовом виде



Рисунок 3.10 – Результат третьего запроса в графическом виде

* + 1. **Запрос 4. Добавление задачи к проекту**

Как упоминалось в разделе 1.5.3, СУБД Neo4j – база данных с гибкой структурой, поэтому запрос на добавление задачи к проекту является простой задачей. Необходимо создать задачу, а затем связать ее с проектом согласно созданной модели. При надобности, имена и значения свойств узла могут меняться.

|  |
| --- |
| MATCH  (p:Project {name:'Audio portal'})  MERGE  (i:Issue{name:'Adding JS in HTML pages'})  MERGE  (p)-[:Consists\_of]->(i) |

Результат такого запроса представлен на рисунке 3.11.

После добавления задачи, запросим у СУБД Neo4j список текущих задач проекта “AUDIO\_PORTAL”, чтобы проверить добавлена ли новая задача.

|  |
| --- |
| MATCH (p:Project)-[:Consists\_of]->(i:Issue)  WHERE p.name = "Audio portal"  RETURN p,i |



Рисунок 3.11 – Результат четвертого запроса

Текущий список задач проекта представлен на рисунке 3.12. Новая задача успешно добавлена и может использоваться для дальнейшей работы. Аналогичными запросами можно добавлять и другие необходимые узлы.



Рисунок 3.12 – Список текущих задач проекта в графическом виде

1. **ОХРАНА ТРУДА**

В текущей выпускной квалификационной работе осуществляется разработка графовой базы данных для системы управления проектами с целью интеграции данных из существующих систем управления проектами. Разработанная графовая база данных увеличит показатели производительности, реализует ссылки на записи в существующих системах управления проектами, а также благодаря своей гибкости, даст возможность изменять модель данных или необходимую ее часть на лету, исходя из требований к интегрированной системе.

В соответствии с федеральным законом «О специальной оценке условий труда» № 426-ФЗ, при изменении технологического процесса, который способен оказать влияние на уровень воздействия вредных и (или) опасных производственных факторов на работников, требуется провести внеплановую специальную оценку условий труда [26]. Проведение внеплановой специальной оценки условий труда подразумевает формирование соответствующей комиссии. Исходя из этого, в текущей ВКР было принято решение воспользоваться «Методикой проведения специальной оценки условий труда» [27].

Согласно федеральному закону «О специальной оценке условий труда» № 426-ФЗ, выделяются следующие гигиенические критерии воздействия факторов рабочей среды и трудового процесса:

1. химический фактор;
2. биологический фактор;
3. аэрозоли преимущественно фиброгенного действия (АПФД);
4. биоакустические факторы;
5. микроклимат;
6. световая среда;
7. неионизирующие электромагнитные поля и излучения;
8. работа с источниками ионизирующих излучений;
9. аэроионный состав воздуха;
10. тяжесть и напряженность трудового процесса.

В виду того, что взаимодействие с разработанной графовой базой данных будут осуществлять системные администраторы и работа с БД слабо влияет на факторы, указанные в пунктах 1-9, будет производиться анализ и оценка условий труда системного администратора по факторам тяжести и напряженности трудового процесса.

* 1. **Оценка тяжести труда системного администратора**

Отнесение условий труда к классу (подклассу) условий труда по тяжести трудового процесса осуществляется по следующим показателям:

1. физическая динамическая нагрузка;
2. масса поднимаемого груза;
3. стереотипные рабочие движения;
4. статическая нагрузка;
5. рабочая поза;
6. наклоны корпуса;
7. перемещение в пространстве.

Указанные показатели тяжести трудового процесса для количественного измерения и оценки требуют своего подхода [27].

**Физическая динамическая нагрузка**

Физическая динамическая нагрузка выражается в единицах внешней механической работы за смену, кг × м.

Чтобы подсчитать физическую динамическую нагрузку (внешнюю механическую работу) необходимо определить массу груза, перемещаемую вручную при каждой операции и путь ее перемещения в метрах. Подсчитывается общее количество операций по переносу груза за смену и суммируется величина внешней механической работы, кг × м, за рабочую смену целиком.

**Масса поднимаемого груза**

Определение массы (кг) груза (поднимаемого или переносимого), требует взвешивания на товарных весах. Регистрируется только максимальная величина. Массу груза можно также определить по документам, прилагаемым к грузу. Чтобы определить суммарную массу груза, перемещаемого в течении каждого часа смены, вес всех грузов суммируется, в случае переносимого груза одного веса, этот вес умножается на число подъёмов или перемещений в течении каждого часа.

**Стереотипные рабочие движения**

В данном контексте, “рабочее движение” трактуется как движение элементарное, то есть однократное перемещение тела или его части из одного положения в другое. Стереотипные рабочие движения в зависимости от нагрузки делятся на локальные и региональные. Для подсчета стереотипных рабочих движений, достаточно подсчитать их количество за рабочую смену.

**Статическая нагрузка**

Статическая нагрузка (кг × с), связанная с поддержанием человеком груза или приложением усилия без перемещения тела или его отдельных частей, рассчитывается путем перемножения двух величин: удерживаемого усилия и времени его удержания.

**Рабочая поза**

Характер рабочей позы (свободная, неудобная, фиксированная, вынужденная) определяется визуально. Время пребывания в вынужденной позе, позе с наклоном корпуса или другой рабочей позе, определяется на основании хронометражных данных за рабочую смену.

**Наклоны корпуса**

Число наклонов за рабочую смену определяется путем их прямого подсчета или определением их количества за одну операцию и умножается на число операций за рабочую смену.

**Перемещение в пространстве**

Перемещение работника в пространстве осуществляется с учетом такого перемещения по горизонтали и (или) вертикали, обусловленного технологическим процессом, в течение рабочей смены и определяется на основании подсчета количества шагов за рабочую смену и измерения длины шага. Количество шагов за рабочий день (смену) определяется с помощью шагомера, помещенного в карман работника или закрепленного на его поясе.

**Сравнительная оценка показателей тяжести трудового процесса до и после разработки графовой БД**

При разработке графовой базы данных изменяются значения тяжести трудового процесса. К таким показателям относятся:

* физическая динамическая нагрузка;
* суммарная масса груза в течении каждого часа смены;
* статическая нагрузка одной рукой;
* стереотипные рабочие движения.

Остальные показатели тяжести трудового процесса останутся без изменений.

При расчете значений показателей тяжести трудового процесса, были определены значения тех величин, которые используются. Системный администратор находится в сидячем положении за рабочим компьютером около 70 процентов времени рабочей смены. Одной из своих рук он перемещает механический манипулятор «компьютерная мышь» на расстоянии 0,10 м, удерживая ее около 0,5 с. Вес компьютерной мыши может различаться от 150 до 200 грамм. При расчете показателей, было принято решение взять средний вес компьютерной мыши, который составил 175 грамм (0,175 кг). Стереотипные движения зависят от конкретной выполняемой системным администратором работы и могут определятся как числом напечатанных символов, так и числом выполненных переносов компьютерной мыши за рабочую смену. Необходимо учитывать, что системный администратор может не выполнять стереотипных движений, ввиду ожидания получения некоторого результата, также системный администратор не всегда печатает символы и при перемещении компьютерной мыши, тратит время на само ее перемещение. Так как характер выполняемых задач системного администратора различен, было принято решение что он совершает порядка 10 000 стереотипных движений за рабочую смену (8 часов) без учета разработанной графовой базы данных.

Чтобы оценить изменения показателей тяжести трудового процесса до и после разработки графовой БД, был произведен расчет их значений при использовании системным администратором реляционной базы данных (MySQL) и разработанной графовой БД (Neo4j). В качестве примера был сформирован запрос на извлечение некоторых данных из этих двух БД. Поиск осуществлялся среди одного миллиона записей. Для реляционной БД, время отклика на запрос заняло около 1,5 минуты, а для разработанной графовой бд – менее 1 минуты. Разработанная графовая БД для системы управления проектами позволяет существенно сократить время отклика на запросы к БД и количество манипуляций при формировании запроса, что в свою очередь сокращает число стереотипных движений. За рабочую смену (8 часов = 480 минут), системный администратор выполняет 10 000 стереотипных движений, но, за счет разработанной графовой БД определяется экономия времени в 30 с (1,5 м – 1 м = 0,5 м = 30 с). Так как за 480 минут системный администратор совершает 10 000 стереотипных движений, что равно 21 стереотипному движению в минуту, определяется экономия в ~10 стереотипных движений. Учитывая, что сложность запросов бывает разная, то и экономия времени может быть разной, как больше, так и меньше. В таблице 7 приведена сравнительная оценка тяжести трудового процесса до и после разработки.

Таблица 7 – Сравнительная оценка тяжести трудового процесса

|  |  |  |
| --- | --- | --- |
| Показатели тяжести трудового процесса | Значение показателя тяжести трудового процесса | |
| До разработки | После разработки |
| Физическая динамическая нагрузка | 0,175 кг × 0,10 м × 10 000 = 175 кг × м – Класс 1 | 0,175 кг × 0,10 м × (10 000 – 4 800) = 91 кг × м – Класс 1 |
| Масса одноразового подъёма груза | 0,175 кг – Класс 1 | 0,175 кг – Класс 1 |

Продолжение таблицы 7

|  |  |  |
| --- | --- | --- |
| Показатели тяжести трудового процесса | Значение показателя тяжести трудового процесса | |
| До разработки | После разработки |
| Суммарная масс груза в течении каждого часа рабочей смены | 0,175 кг × 10 000 = 1 750 кг и разделить на 8 ч работы в смену = 218,75 кг – Класс 1 | 0,175 кг × (10 000 – 4 800) = 910 кг и разделить на 8 ч работы в смену = 113,75 кг – Класс 1 |
| Стереотипные движения (региональная нагрузка на мышцы рук и плечевого пояса) | 10 000 – Класс 1 | 10 000 – 4 800 = 5 200 – Класс 1 |
| Статическая нагрузка одной рукой | 0,175 кг × 0,5 с = 0,0875 кг × с, так как компьютерная мышь удерживается в течении 0,5 с. Статическая нагрузка за рабочую смену одной рукой: 0,0875 кгс × 10 000 = 875 кг × с – Класс 1 | 0,175 кг × 0,5 с = 0,0875 кг × с, так как компьютерная мышь удерживается в течении 0,5 с. Статическая нагрузка за рабочую смену одной рукой: 0,0875 кгс × (10 000 – 4 800) = 455 кг × с – Класс 1 |
| Наклоны корпуса за смену | Количество наклонов за рабочую смену – до 50 – Класс 1 | Количество наклонов за рабочую смену – до 50 – Класс 1 |

Продолжение таблицы 7

|  |  |  |
| --- | --- | --- |
| Показатели тяжести трудового процесса | Значение показателя тяжести трудового процесса | |
| До разработки | После разработки |
| Рабочая поза | Периодическое, до 25% времени смены, нахождение в неудобной (работа с поворотом туловища, неудобным размещением конечностей и др.) и/или фиксированной позе (невозможность изменения взаимного положения различных частей тела относительно друг друга) – Класс 2 | Периодическое, до 25% времени смены, нахождение в неудобной (работа с поворотом туловища, неудобным размещением конечностей и др.) и/или фиксированной позе (невозможность изменения взаимного положения различных частей тела относительно друг друга) – Класс 2 |
| Перемещение в пространстве, обусловленное технологическим процессом | Системный администратор, в основном, сидит за своим рабочим местом, перемещения незначительные, до 1 км за рабочую смену – Класс 1 | Системный администратор, в основном, сидит за своим рабочим местом, перемещения незначительные, до 1 км за рабочую смену – Класс 1 |

Итоговая оценка тяжести трудового процесса приравнивается к показателю, который отнесен к наибольшему классу. Наибольший класс условий труда системного администратора является – 2 (Допустимый). Исходя из этого, общая тяжесть трудового процесса системного администратора имеет «Допустимый» класс.

* 1. **Оценка напряженности труда системного администратора**

Разработанная графовая база данных для системы управления проектами не влияет на показатели напряженности трудового процесса.

Отнесение условий труда к классу (подклассу) условий труда по напряженности трудового процесса осуществляется по следующим показателям:

1. плотность сигналов и сообщений;
2. число производственных объектов одновременного наблюдения;
3. работа с оптическими приборами;
4. нагрузка на голосовой аппарат;
5. монотонность нагрузок.

Указанные показатели напряженности трудового процесса для количественного измерения и оценки требуют своего подхода [27].

**Плотность сигналов и сообщений**

Плотность сигналов и сообщений в среднем за 1 час рабочей смены рассчитывается путем подсчета количества воспринимаемых и передаваемых сигналов, сообщений и распоряжений.

**Число производственных объектов одновременного наблюдения**

Число производственных объектов одновременного наблюдения подсчитывается путем оценки объёма внимания (от 4 до 8 несвязанных объектов) и его распределения (способности одновременно сосредотачивать внимание на нескольких объектах или действиях).

**Работа с оптическими приборами**

Оценка работы с оптическими приборами рассчитывается в процентном соотношении от продолжительности рабочей смены и осуществляется на основе хронометражных наблюдений.

**Нагрузка на голосовой аппарат**

Нагрузка на голосовой аппарат работника рассчитывается в суммарном количестве часов, наговариваемых в неделю и осуществляется с учетом продолжительности речевых нагрузок на основе хронометражных наблюдений.

**Монотонность нагрузок**

Оценка монотонности нагрузок осуществляется с учетом числа элементов (приемов), необходимых для реализации простого задания или многократно повторяющихся операции, и продолжительности выполнения простых производственных заданий или повторяющихся операций, времени активных действий, монотонности производственной обстановки.

Оценка напряженности трудового процесса системного администратора представлена в таблице 8.

Таблица 8 – Оценка напряженности труда

|  |  |  |
| --- | --- | --- |
| Показатели напряженности трудового процесса | Значение показателя напряженности трудового процесса | Класс условий труда |
| Сенсорные нагрузки | | |
| Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы | До 75 | 1 |
| Число производственных объектов одновременного наблюдения | До 5 | 1 |
| Работа с оптическими приборами (% времени рабочей смены) | До 25 | 1 |

Продолжение таблицы 8

|  |  |  |
| --- | --- | --- |
| Показатели напряженности трудового процесса | Значение показателя напряженности трудового процесса | Класс условий труда |
| Сенсорные нагрузки | | |
| Нагрузка на голосовой аппарат (суммарное количество часов, наговариваемое в неделю) | До 16 | 1 |
| Монотонность нагрузок | | |
| Число элементов (приемов), необходимых для реализации задания | До 10 | 2 |
| Время пассивного наблюдения за ходом техпроцесса в % от времени рабочей смены | Менее 75 | 1 |

Итоговая оценка напряженности трудового процесса приравнивается к показателю, который отнесен к наибольшему классу. Наибольший класс условий труда является – 2 (Допустимый). Исходя из этого, общая напряженность трудового процесса системного администратора имеет «Допустимый» класс.

* 1. **Выводы по разделу**

В данном разделе выполнены оценка и анализ тяжести и напряжённости трудового процесса системного администратора.

В текущей выпускной квалификационной работе, разработанная графовая база данных для системы управления проектами не меняет класс условий труда системного администратора, но некоторые показатели тяжести труда улучшились за счет уменьшения времени отклика на запросы к БД и количества манипуляций при формировании запроса. Разработанная графовая БД позволила сократить количество стереотипных движений, что привело к экономии времени и соответственно к уменьшению совершаемых движений. Таким образом, улучшились следующие показатели тяжести трудового процесса: физическая динамическая нагрузка с 175 кг × м до 91 кг × м, суммарная масса груза в течении каждого часа рабочей смены с 218,75 кг до 113,75 кг, стереотипные движения с 10 000 до 5 200 движений, статическая нагрузка одной рукой с 875 кг × с до 455 кг × с за рабочую смену. Разработанная графовая база данных для системы управления проектами не повлияла на показатели напряженности трудового процесса.

1. **ОБОСНОВАНИЕ ЭКНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ**

В текущей выпускной квалификационной работе осуществляется разработка графовой базы данных для системы управления проектами с целью интеграции данных из существующих систем управления проектами.

Разработка графовой базы данных увеличит показатели производительности, реализует ссылки на записи в существующих системах управления проектами, а также благодаря своей гибкости, даст возможность изменять модель данных или необходимую ее часть на лету, исходя из требований к интегрированной системе. Предоставляемая производительность разработанной графовой БД существенно снизит время отклика БД на запросы пользователей интегрированной системы, что позволит сократить расход материальных средств на обслуживание и мониторинг базы данных.

* 1. **План проведения работ**

Что бы рассчитать общее время, затраченное на выполнение работы, требуется проанализировать календарный план, который представлен в таблице 9.

Таблица 9 – Календарный план времени написания ВКР

|  |  |  |
| --- | --- | --- |
| № п/п | Этапы ВКР | Затраченное время, дней |
| 1 | Постановка задачи | 1 |
| 2 | Разработка календарного плана | 1 |
| 3 | Подбор необходимой литературы | 2 |
| 4 | Изучение жизненного цикла проекта | 1 |
| 5 | Ознакомление с системами управления проектами | 1 |
| 6 | Анализ исходного перечня систем управления проектами | 1 |
| 7 | Ознакомление с графовой базой данных Neo4j | 2 |

Продолжение таблицы 9

|  |  |  |
| --- | --- | --- |
| № п/п | Этапы ВКР | Затраченное время, дней |
| 8 | Анализ отличий реляционных баз данных от графовых | 2 |
| 9 | Разработка графовой базы данных | 10 |
| 10 | Изучение языка запросов Cypher | 2 |
| 11 | Создание скриптов для заполнения графовой БД и запросов для извлечения информации | 2 |
| 12 | Составление пояснительной записки | 9 |
| 13 | Написание раздела «Охрана труда» | 3 |
| 15 | Оценка экономической эффективности разработки | 3 |

В календарный план не вошли такие этапы, как сдача разделов на проверку консультантам, поскольку эти этапы лишены трудозатрат. Исходя из этого, затраченное время на выполнение работы составило 40 дней. Так как средняя продолжительность рабочего дня составляет 8 часов, общее затраченное время равно 320 часам.

* 1. **Смета затрат на выпускную квалификационную работу**

Для разработки графовой базы данных системы управления проектами, представлена смета затрат по следующим статьям калькуляции:

1. материалы, покупные изделия и полуфабрикаты;
2. связь, источники информации;
3. расходы на электроэнергию;
4. заработная плата;
5. амортизация оборудования и программного обеспечения.
   * 1. **Материалы, покупные изделия и полуфабрикаты**

Необходимо рассчитать стоимость покупных изделий, которые требуются для выполнения ВКР.

Список покупных изделий:

1. бумага формата А4;
2. чернила для принтера (HP GT51);
3. папка;
4. канцелярские принадлежности (ручка, карандаш, блокнот).

Стоимость материалов и покупных изделий рассчитана по формуле (5.1).

|  |  |
| --- | --- |
|  | (5.1) |

*где*

**С** – стоимость изделия за штуку, руб.;

**К** – количество изделий, шт.;

**Ц** – цена изделия, руб.

В таблице 10 представлен расчет стоимости покупных изделий.

Таблица 10 – Стоимость материалов и покупных изделий

|  |  |  |  |
| --- | --- | --- | --- |
| Изделие | Количество | Цена за единицу, руб. | Стоимость, руб. |
| Бумага А4 | 1 пачка | 210 | 210 |
| HP GT51 | 1 шт. | 690 | 690 |
| Папка | 1 шт. | 180 | 180 |
| Ручка | 2 шт. | 40 | 80 |
| Карандаш | 2 шт. | 25 | 50 |
| Блокнот | 1 шт. | 50 | 50 |
| ИТОГО: | | | 1 260 |

* + 1. **Связь, источники информации**

Основным источником информации и средством коммуникации был интернет, на оплату которого было затрачено 500р. в месяц. Также была приобретена книга по графовым базам данных, стоимость которой составило 970р.

Затраты за время разработки рассчитаны по формуле (5.2).

|  |  |
| --- | --- |
|  | (5.2) |

*где*

**ЗИ** – затраты на источники информации, руб.;

**СМ** – месячная стоимость интернета, руб.;

**nМ** – количество месяцев на разработку, един, мес.;

**СК** – стоимость приобретённой книги, руб.

Таким образом, .

* + 1. **Расходы на электроэнергию**

Расходы на электроэнергию рассчитаны по следующей формуле (5.3).

|  |  |
| --- | --- |
|  | (5.3) |

*где*

**ЗЭi** – затраты на электроэнергию i-оборудованием, руб.;

**QЭi** – количество электроэнергии потребляемой i-оборудованием в час, кВт;

**ЦЭ** – стоимость одного кВт/ч, руб.

По формуле (5.4) рассчитано количество электроэнергии, потребляемой i-оборудованием.

|  |  |
| --- | --- |
|  | (5.4) |

*где*

**QЭi** – количество электроэнергии, потребляемой i-оборудованием, кВт;

**Ni** – мощность используемого i-оборудования, кВт;

**Ti** – длительность расчетного периода, ч;

**ni** – количество оборудования, шт. (единиц).

Время работы над ВКР – 1 месяц и 10 дней (40 дней). Работа велась каждый день, в среднем по 8 часов. Потребляемая мощность ноутбука составляет 100 Вт. Расчет значения длительности работы оборудования составил 320 часов.

Общее количество электроэнергии потребляемой ноутбуком составило 32 кВт/ч.

Потребляемая мощность принтера составляет 190 Вт. Поскольку печатная версия пояснительной записки была распечатана в нескольких экземплярах, а также печатались дополнительные необходимые документы, общее время работы принтера составило примерно 2,5 часа.

Результаты расхода электроэнергии представлены в таблице 11. Ставка тарифов на электроэнергию по г. Санкт-Петербург взята на первое полугодие 2017 года.

Таблица 11 – Расходы на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оборудование | Расход электроэнергии, кВт-ч | Цена за единицу, руб./кВт-ч | Длительность работы оборудования, ч | Сумма, руб. |
| Ноутбук | 0,1 | 4,29 | 320 | 137,28 |
| Принтер | 0,19 | 4,29 | 2,5 | 2,03 |
| ИТОГО: | | | | 139,31 |

* + 1. **Расходы на оплату труда**

Расходы на оплату труда состоят из:

1. стипендии студента;
2. зарплаты дипломного руководителя;
3. зарплаты консультанта кафедры «Безопасность жизнедеятельности»;
4. зарплаты консультанта кафедры «Экономика транспорта».

Сумма затрат на выплату заработной платы состоит из стипендии студента и заработной платы консультантов. Расходы на оплату труда рассчитаны по формуле (5.5).

|  |  |
| --- | --- |
|  | (5.5) |

*где*

**СТР** – расходы на оплату труда, руб.;

**З/П** – заработная плата работника, руб.;

**М** – рабочие часы в месяц, 176 ч;

**tр** – время, затраченное на работу, ч.

Расчет расходов на оплату труда представлен в таблице 12.

Таблица 12 – Расходы на оплату труда

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель | Заработная плата, руб./мес. | Затраченное время, ч. | Расходы на оплату труда, руб. |
| Дипломник | 2 226 | 320 | 4 047,3 |
| Дипломный руководитель | 30 000 | 20 | 3 409,1 |
| Консультант по охране труда | 25 000 | 2 | 284,1 |
| Консультант по экономике | 25 000 | 2 | 284,1 |
| ИТОГО: | | | 8 024,6 |

* + 1. **Расходы на амортизацию оборудования**

Амортизационные отчисления (АО) – отчисления части стоимости основных фондов для возмещения их износа. Амортизационные отчисления включаются в издержки производства. Производятся на основе норм и балансовой стоимости основных фондов, на которые начисляется амортизация [28].

Требуется рассчитать амортизацию по каждому оборудованию и программному продукту. В таблице 13 указана исходная стоимость оборудования и программного обеспечения, а также их амортизационный период.

Амортизационный период оборудования АП в настоящее время равен сроку морального старения и составляет 3 года. Исходя из этого, за 3 года амортизация оборудования равна стоимости оборудования.

Расходы на амортизационные отчисления для оборудования и программного обеспечения рассчитаны по формуле (5.6).

|  |  |
| --- | --- |
|  | (5.6) |

*где*

**АО** – амортизационные отчисления, руб.;

**С** – стоимость оборудования, руб.;

**АП** – амортизационный период оборудования, мес.;

**Т** – время пользования оборудования, мес.

Таблица 13 – Исходная стоимость оборудования и ПО

|  |  |  |
| --- | --- | --- |
| Наименование оборудования или ПО | Амортизационный период | Стоимость, руб. |
| Ноутбук Acer Aspire E5-575G-33V1 | 3 года | 43 800 |
| Струйный принтер Canon PIXMA IP7240 | 3 года | 4 990 |
| Компьютерная мышь Hewlett Packard X3500 | 3 года | 1 190 |
| Операционная система Windows 8.1 | 3 года | 6 999 |
| Microsoft Office 2010 | 3 года | 6 580 |
| ИТОГО: | | 63 559 |

Отталкиваясь от приведенной в таблице 13 стоимости оборудования и программного обеспечения, амортизационные отчисления составили:

* 1. **Стоимость выполнения ВКР**

Стоимость выполнения выпускной квалификационной работы является суммой всех расходов, перечисленных выше. Калькуляция по статьям расходов и итоговая стоимость ВКР представлена в таблице 14.

Таблица 14 – Итоговая стоимость ВКР

|  |  |  |
| --- | --- | --- |
| № п/п | Статья расходов | Сумма, руб. |
| 1 | Материалы, покупные изделия и полуфабрикаты | 1 260 |
| 2 | Связь, источники информации | 1 970 |
| 3 | Расходы на электроэнергию | 139,31 |
| 4 | Заработная плата | 8 024,6 |
| 5 | Амортизация оборудования и программного обеспечения | 2 295 |
| ИТОГО: | | 13 688,91 |

* 1. **Оценка эффективности разработки**

Разработанная графовая база данных для системы управления проектами предоставляет высокую производительность, которая заключается в уменьшении времени отклика БД на запросы пользователей интегрированной системы как минимум в 2 раза (50%). Такая производительность достигается за счет графовой модели данных, которая локализует запросы в той части графа, в которой находятся необходимые данный, при этом время отклика зависит только от размера части графа, которую требуется обойти для удовлетворения запроса, а не от общего размера графа.

* + 1. **Расчет экономии времени**

Среднее количество времени, потраченное системным администратором на мониторинг и обслуживание базы данных при возникновении конфликтных ситуации выполнения запросов, составляет примерно 5% рабочего времени.

Время, сэкономленное за счет уменьшения времени отклика на запросы пользователей (из расчета на 1 месяц) рассчитано по формуле (5.7).

|  |  |
| --- | --- |
|  | (5.7) |

*где*

**Т** – сэкономленное время, час/мес.;

**М** – рабочие часы в месяц, 176 ч;

**КОТКЛ** – коэффициент производительности времени отклика, 0,5;

**КРАБ** – коэффициент затраченного рабочего времени, 0,05.

Таким образом, сэкономленное время составляет:

час/мес.

* + 1. **Расчет экономии средств**

Средняя заработная плата системного администратора составляет 50 000 рублей. Количество сэкономленных средств за один месяц рассчитано по формуле (5.8).

|  |  |
| --- | --- |
|  | (5.8) |

*где*

**КСЭ** – количество сэкономленных средств, руб./мес.;

**З/П** – заработная плата системного администратора, руб./мес.;

**КОТКЛ** – коэффициент производительности времени отклика, 0,5;

**КРАБ** – коэффициент затраченного рабочего времени, 0,05.

Таким образом, количество сэкономленных средств составляет:

* + 1. **Расчет окупаемости разработки**

Время, за которое окупится разработанная графовая БД для системы управления проектами в текущей выпускной квалификационной работе вычислено по формуле (5.9).

|  |  |
| --- | --- |
|  | ((5.9) |

*где*

**ТОК** – время окупаемости, мес.;

**СВКР** – стоимость ВКР, руб.;

**КСЭ** – количество сэкономленных средств, руб./мес.

Рассчитанное время окупаемости разработки составляет:

* 1. **Выводы по разделу**

В данном разделе произведена калькуляция расходов на разработку графовой базы данных для системы управления проектами, а также оценена эффективность разработки, которая заключается в экономии времени и средств за счет уменьшения времени отклика БД на запросы пользователей интегрированной системы.

При расчете стоимости, учитывались расходы на материалы, связь, источники информации и расходы на электроэнергию. Определены расходы на оплату труда дипломника, дипломного руководителя и консультантов. Также были рассчитаны затраты на амортизацию оборудования. Общая стоимость ВКР составила 13 688,91 руб.

Поскольку разработанная графовая БД должна сократить время отклика на запросы пользователей интегрированной системы, эксплуатация такой БД несет определенную выгоду, а именно время, сэкономленное системным администратором на мониторинге и обслуживании БД при возникновении конфликтных ситуаций выполнения запросов. Сэкономленное время планируется потратить на другую работу, связанную с системным администрированием. Период окупаемости разработки составил приблизительно 11 месяцев.

**ЗАКЛЮЧЕНИЕ**

В рамках текущей выпускной квалификационной работы разработана графовая база данных для системы управления проектами. Разработанная модель данных предоставляет возможность интегрирования записей из существующих систем управления проектами. Также были реализованы скрипты для заполнения графовой БД и созданы запросы для извлечения основной информации о проектах. Разработанная графовая модель данных является гибкой и может изменяться в зависимости от требований к интегрированной системе. Ввиду использования графовой СУБД Neo4j, разработанная база данных обеспечивает высокую производительность и эффективное масштабирование.

Дальнейшее развитие работы предполагает разработку веб-интерфейса и API для интегрирования проектов других систем управления проектами в разработанную графовую базу данных.

В разделе «Охрана труда» была проведена оценка условий труда по факторам тяжести и напряженности трудового процесса до и после разработанной графовой базы данных.

В разделе «Обоснование экономической эффективности» была проведена калькуляция расходов на разработку графовой БД и оценена эффективность данной разработки. В результате калькуляции, себестоимость выполнения ВКР составляет 13 688,91 рублей, а экономия при использовании разработки составляет 1 250 рублей в месяц. Исходя из этого, расчетный период окупаемости ВКР составляет 11 месяцев, что является благоприятным сроком.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Ричард Ньютон. Управление проектами от А до Я / Пер. с англ. Кириченко А.Р. – Москва: Изд-во Альпина Бизнес Букс, 2007. – 36 с.
2. Боронина Л.Н. Основы управления проектами: [учебное пособие] / Л. Н. Боронина, З. В. Сенук; М-во образования и науки Рос. Федерации, Урал. федер. ун-т. – Екатеринбург: Изд-во Урал. ун-та, 2015. – 112 с.
3. Институт управления проектами США. PMBOK: Руководство к Своду знаний по управлению проектами, 4-е изд., PMI, 2008. – 241 с.
4. ЭУП: Электронное учебное пособие [Электронный ресурс] – Режим доступа: <http://eos.ibi.spb.ru/umk/11_18/5/5_R0_T3.html>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
5. Библиотека Online. Лекции по управлению программными проектами [Электронный ресурс] – Режим доступа: <http://citforum.ru/SE/project/arkhipenkov_lectures/5.shtml>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
6. Горбовцов Г.Я. Система управления проектами: [учебное пособие] / Г.Я. Горбовцов. – Москва: Изд. центр ЕАОИ, 2011. – 344 с.
7. GitHub [Электронный ресурс] – Режим доступа: <https://github.com/about>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
8. Bitbucket [Электронный ресурс] – Режим доступа: <https://ru.atlassian.com/software/bitbucket>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
9. Jira [Электронный ресурс] – Режим доступа: <https://ru.atlassian.com/software/jira>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
10. GitLab [Электронный ресурс] – Режим доступа: <https://about.gitlab.com/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
11. Git [Электронный ресурс] – Режим доступа: <https://git-scm.com/about/small-and-fast>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
12. Д. Кренке. Теория и практика построения баз данных. 9-е изд. / Д. Кренке. – Санкт-Петербург: Изд-во Питер, 2005. – 800 с.
13. Техническая документации Майкрософт [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/azure/documentdb/documentdb-nosql-vs-sql>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
14. Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. Под редакцией Жаклин Картер / пер. с англ. Слинкин А.А. – Москва: Изд-во ДМК Пресс, 2013. – 384 с.
15. Робинсон Ян, Вебер Джим, Эифрем Эмиль. Графовые базы данных: новые возможности для работы со связанными данными / пер. с англ. Рагимова Р.Н.; науч. ред. Киселёв А.Н. – 2-е изд. – Москва: Изд-во ДМК Пресс, 2016. – 256 с.
16. Издательский дом "Вильямс": Более подробно о моделях данных [Электронный ресурс] – Режим доступа: <http://www.williamspublishing.com/PDF/978-5-8459-1829-1/part.pdf>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
17. IBM: Processing large-scale graph data: A guide to current technology [Электронный ресурс] – Режим доступа: <https://www.ibm.com/developerworks/library/os-giraph/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
18. Cassovary [Электронный ресурс] – Режим доступа: <https://github.com/twitter/cassovary>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
19. Pegasus [Электронный ресурс] – Режим доступа: <http://www.cs.cmu.edu/~pegasus/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
20. Apache Giraph [Электронный ресурс] – Режим доступа: <http://giraph.apache.org/intro.html>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
21. ACM Digital Library [Электронный ресурс] – Режим доступа: <http://dl.acm.org/citation.cfm?id=1807184>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
22. What is Neo4j? [Электронный ресурс] – Режим доступа: <https://neo4j.com/developer/graph-database/#_what_is_neo4j>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
23. Дюбуа П. MySQL. Сборник рецептов / пер. с англ. Шера П.Н. – Санкт-Петербург: Изд-во Символ‑Плюс, 2006. – 356 с.
24. Ресурс для IT-специалистов. MySQL vs Neo4j [Электронный ресурс] – Режим доступа: <https://habrahabr.ru/post/258179/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
25. Chapter 3. Cypher [Электронный ресурс] – Режим доступа: <http://neo4j.com/docs/developer-manual/current/cypher/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
26. Федеральный закон от 28.12.2013 г. №426-ФЗ «О специальной оценке условий труда».
27. КонсультантПлюс. Методика проведения специальной оценки условий труда [Электронный ресурс] – Режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_158398/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).
28. КонсультантПлюс. Амортизационные отчисления [Электронный ресурс] – Режим доступа: <http://www.consultant.ru/document/cons_doc_LAW_51295/>, свободный. – Загл. с экрана (дата обращения: 04.06.2017).

**ПРИЛОЖЕНИЕ А**

Скрипт для проекта “DS\_IO\_API”.

Код скрипта:

CREATE

//Создание вершин, связанных с описанием проекта--------------------------------

(p1:Project {name:'DS IO API'}),

(p2:Customer {firstname: 'Bob',lastname: 'Marley'}),

(p3:Project\_type {type:'API project'}),

(p4:Project\_code {code:'F002'}),

(p5:Project\_status {status:'Closed'}),

//------------------------------------------------------------------------------

//Вершина MASTER----------------------------------------------------------------

(m:Master {firstname:'Roman',lastname:'Lobov'}),

//------------------------------------------------------------------------------

//Вершины DEVELOPERS------------------------------------------------------------

(d1:Developer {firstname:'Dmitry',lastname:'Kungurov'}),

(d2:Developer {firstname:'Kamil',lastname:'Hasanov'}),

(d3:Developer {firstname:'Vladislav',lastname:'Karakozov'}),

//------------------------------------------------------------------------------

//Вершины ISSUES----------------------------------------------------------------

(d11:Issue {name:'Defining non-VSAM data set types'}),

(d12:Issue {name:'Call other modules'}),

(d13:Issue {name:'Parsing parameters'}),

(d14:Issue {name:'Defining VSAM data set types'}),

(d15:Issue {name:'Creating description file'}),

//------------------------------------------------------------------------------

(d21:Issue {name:'Open function for DA module'}),

(d22:Issue {name:'Close function for DA module'}),

(d23:Issue {name:'SEEK function for DA module'}),

(d24:Issue {name:'READ function for DA module'}),

(d25:Issue {name:'WRITE function for DA module'}),

//------------------------------------------------------------------------------

(d31:Issue {name:'OPEN function for KSDS module'}),

(d32:Issue {name:'READ function for KSDS module'}),

(d33:Issue {name:'WRITE function for KSDS module'}),

(d34:Issue {name:'CREATE function for KSDS nodule'}),

(d35:Issue {name:'CLOSE function for KSDS module'}),

//------------------------------------------------------------------------------

//Вершины REPOSITORY------------------------------------------------------------

(l1:Repository {name:'GitLab', URL:'https://gitlab.com/Kungurov/DSIOAPI'}),

(l2:Repository {name:'Bitbucket', URL:'https://bitbucket.org/Hasanov/dsioapi'}),

(l3:Repository {name:'GitHub', URL:'https://github.com/Karakozov/DsIOAPI'}),

(l4:Repository {name:'GitHub', [URL:'https://github.com/Lobov/DsIOAPI'}](URL:'https://github.com/Lobov/DsIOAPI'%7d)),

//------------------------------------------------------------------------------

//Вершины описания ISSUES-------------------------------------------------------

(d11a:Issue\_type {type:'Sub task'}),

(d11b:Issue\_status {status:'Done'}),

(d11c:Issue\_priority {priority:'Low'}),

//------------------------------------------------------------------------------

(d12a:Issue\_type {type:'Task'}),

(d12b:Issue\_status {status:'Done'}),

(d12c:Issue\_priority {priority:'Medium'}),

//------------------------------------------------------------------------------

(d13a:Issue\_type {type:'Task'}),

(d13b:Issue\_status {status:'Done'}),

(d13c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d14a:Issue\_type {type:'Sub task'}),

(d14b:Issue\_status {status:'Done'}),

(d14c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d15a:Issue\_type {type:'Task'}),

(d15b:Issue\_status {status:'Done'}),

(d15c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d21a:Issue\_type {type:'New feature'}),

(d21b:Issue\_status {status:'Done'}),

(d21c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d22a:Issue\_type {type:'New feature'}),

(d22b:Issue\_status {status:'Done'}),

(d22c:Issue\_priority {priority:'Medium'}),

//------------------------------------------------------------------------------

(d23a:Issue\_type {type:'New feature'}),

(d23b:Issue\_status {status:'Done'}),

(d23c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d24a:Issue\_type {type:'New feature'}),

(d24b:Issue\_status {status:'Done'}),

(d24c:Issue\_priority {priority:'Medium'}),

//------------------------------------------------------------------------------

(d25a:Issue\_type {type:'New feature'}),

(d25b:Issue\_status {status:'Done'}),

(d25c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d31a:Issue\_type {type:'New feature'}),

(d31b:Issue\_status {status:'Done'}),

(d31c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d32a:Issue\_type {type:'New feature'}),

(d32b:Issue\_status {status:'Done'}),

(d32c:Issue\_priority {priority:'Medium'}),

//------------------------------------------------------------------------------

(d33a:Issue\_type {type:'New feature'}),

(d33b:Issue\_status {status:'Done'}),

(d33c:Issue\_priority {priority:'Medium'}),

//------------------------------------------------------------------------------

(d34a:Issue\_type {type:'New feature'}),

(d34b:Issue\_status {status:'Done'}),

(d34c:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(d35a:Issue\_type {type:'New feature'}),

(d35b:Issue\_status {status:'Done'}),

(d35c:Issue\_priority {priority:'Low'}),

//------------------------------------------------------------------------------

//Создание отношений между вершинами / LINKS BETWEEN NODES----------------------

(p2)-[:Orders]->(p1),

(p1)-[:Type\_of\_project]->(p3),

(p1)-[:Code]->(p4),

(p1)-[:Status\_of\_project]->(p5),

(p1)-[:Consists\_of]->(d11),

(p1)-[:Consists\_of]->(d12),

(p1)-[:Consists\_of]->(d13),

(p1)-[:Consists\_of]->(d14),

(p1)-[:Consists\_of]->(d15),

(p1)-[:Consists\_of]->(d21),

(p1)-[:Consists\_of]->(d22),

(p1)-[:Consists\_of]->(d23),

(p1)-[:Consists\_of]->(d24),

(p1)-[:Consists\_of]->(d25),

(p1)-[:Consists\_of]->(d31),

(p1)-[:Consists\_of]->(d32),

(p1)-[:Consists\_of]->(d33),

(p1)-[:Consists\_of]->(d34),

(p1)-[:Consists\_of]->(d35),

//------------------------------------------------------------------------------

(m)-[:Defines]->(p3),

(m)-[:Assigns]->(p4),

(m)-[:Sets]->(p5),

(m)-[:Creates]->(p1),

(m)-[:Appoints]->(d1),

(m)-[:Appoints]->(d2),

(m)-[:Appoints]->(d3),

(m)-[:Creates]->(d11),

(m)-[:Creates]->(d12),

(m)-[:Creates]->(d13),

(m)-[:Creates]->(d14),

(m)-[:Creates]->(d15),

(m)-[:Creates]->(d21),

(m)-[:Creates]->(d22),

(m)-[:Creates]->(d23),

(m)-[:Creates]->(d24),

(m)-[:Creates]->(d25),

(m)-[:Creates]->(d31),

(m)-[:Creates]->(d32),

(m)-[:Creates]->(d33),

(m)-[:Creates]->(d34),

(m)-[:Creates]->(d35),

//------------------------------------------------------------------------------

(m)-[:Defines]->(d11a),

(m)-[:Assigns]->(d11c),

(m)-[:Defines]->(d12a),

(m)-[:Assigns]->(d12c),

(m)-[:Defines]->(d13a),

(m)-[:Assigns]->(d13c),

(m)-[:Defines]->(d14a),

(m)-[:Assigns]->(d14c),

(m)-[:Defines]->(d15a),

(m)-[:Assigns]->(d15c),

(m)-[:Defines]->(d21a),

(m)-[:Assigns]->(d21c),

(m)-[:Defines]->(d22a),

(m)-[:Assigns]->(d22c),

(m)-[:Defines]->(d23a),

(m)-[:Assigns]->(d23c),

(m)-[:Defines]->(d24a),

(m)-[:Assigns]->(d24c),

(m)-[:Defines]->(d25a),

(m)-[:Assigns]->(d25c),

(m)-[:Defines]->(d31a),

(m)-[:Assigns]->(d31c),

(m)-[:Defines]->(d32a),

(m)-[:Assigns]->(d32c),

(m)-[:Defines]->(d33a),

(m)-[:Assigns]->(d33c),

(m)-[:Defines]->(d34a),

(m)-[:Assigns]->(d34c),

(m)-[:Defines]->(d35a),

(m)-[:Assigns]->(d35c),

//------------------------------------------------------------------------------

(d1)-[:Executes\_the]->(d11),

(d1)-[:Executes\_the]->(d12),

(d1)-[:Executes\_the]->(d13),

(d1)-[:Executes\_the]->(d14),

(d1)-[:Executes\_the]->(d15),

//------------------------------------------------------------------------------

(d2)-[:Executes\_the]->(d21),

(d2)-[:Executes\_the]->(d22),

(d2)-[:Executes\_the]->(d23),

(d2)-[:Executes\_the]->(d24),

(d2)-[:Executes\_the]->(d25),

//------------------------------------------------------------------------------

(d3)-[:Executes\_the]->(d31),

(d3)-[:Executes\_the]->(d32),

(d3)-[:Executes\_the]->(d33),

(d3)-[:Executes\_the]->(d34),

(d3)-[:Executes\_the]->(d35),

//------------------------------------------------------------------------------

(d1)-[:Sets]->(d11b),

(d1)-[:Sets]->(d12b),

(d1)-[:Sets]->(d13b),

(d1)-[:Sets]->(d14b),

(d1)-[:Sets]->(d15b),

(d2)-[:Sets]->(d21b),

(d2)-[:Sets]->(d22b),

(d2)-[:Sets]->(d23b),

(d2)-[:Sets]->(d24b),

(d2)-[:Sets]->(d25b),

(d3)-[:Sets]->(d31b),

(d3)-[:Sets]->(d32b),

(d3)-[:Sets]->(d33b),

(d3)-[:Sets]->(d34b),

(d3)-[:Sets]->(d35b),

//------------------------------------------------------------------------------

(d11)-[:Priority]->(d11c),

(d11)-[:Status\_of\_issue]->(d11b),

(d11)-[:Type\_of\_issue]->(d11a),

(d11)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d12)-[:Priority]->(d12c),

(d12)-[:Status\_of\_issue]->(d12b),

(d12)-[:Type\_of\_issue]->(d12a),

(d12)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d13)-[:Priority]->(d13c),

(d13)-[:Status\_of\_issue]->(d13b),

(d13)-[:Type\_of\_issue]->(d13a),

(d13)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d14)-[:Priority]->(d14c),

(d14)-[:Status\_of\_issue]->(d14b),

(d14)-[:Type\_of\_issue]->(d14a),

(d14)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d15)-[:Priority]->(d15c),

(d15)-[:Status\_of\_issue]->(d15b),

(d15)-[:Type\_of\_issue]->(d15a),

(d15)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d21)-[:Priority]->(d21c),

(d21)-[:Status\_of\_issue]->(d21b),

(d21)-[:Type\_of\_issue]->(d21a),

(d21)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d22)-[:Priority]->(d22c),

(d22)-[:Status\_of\_issue]->(d22b),

(d22)-[:Type\_of\_issue]->(d22a),

(d22)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d23)-[:Priority]->(d23c),

(d23)-[:Status\_of\_issue]->(d23b),

(d23)-[:Type\_of\_issue]->(d23a),

(d23)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d24)-[:Priority]->(d24c),

(d24)-[:Status\_of\_issue]->(d24b),

(d24)-[:Type\_of\_issue]->(d24a),

(d24)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d25)-[:Priority]->(d25c),

(d25)-[:Status\_of\_issue]->(d25b),

(d25)-[:Type\_of\_issue]->(d25a),

(d25)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d31)-[:Priority]->(d31c),

(d31)-[:Status\_of\_issue]->(d31b),

(d31)-[:Type\_of\_issue]->(d31a),

(d31)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d32)-[:Priority]->(d32c),

(d32)-[:Status\_of\_issue]->(d32b),

(d32)-[:Type\_of\_issue]->(d32a),

(d32)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d33)-[:Priority]->(d33c),

(d33)-[:Status\_of\_issue]->(d33b),

(d33)-[:Type\_of\_issue]->(d33a),

(d33)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d34)-[:Priority]->(d34c),

(d34)-[:Status\_of\_issue]->(d34b),

(d34)-[:Type\_of\_issue]->(d34a),

(d34)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

(d35)-[:Priority]->(d35c),

(d35)-[:Status\_of\_issue]->(d35b),

(d35)-[:Type\_of\_issue]->(d35a),

(d35)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

//Изменения / COMMITS D11 (Dmitry Kungurov) ------------------------------------

(kd111:Commit {Id:'xbn4509', Author:'Dmitry Kungurov', Message:'Abend in DFHEIIC when runninx in AMODE was fixed', Date:'2016-09-24'}),

(kd111)-[:Link\_to]->(l1),

(kd112:Commit {Id:'9356nm1', Author:'Dmitry Kungurov', Message:'Fixed an invalid record format was requested for a SYSIN and SYSOUT dataset', Date:'2016-09-25'}),

(kd112)-[:Link\_to]->(l1),

(kd113:Commit {Id:'jkl3409', Author:'Dmitry Kungurov', Message:'Fixed QISAM I/O ERROR', Date:'2016-09-27'}),

(kd113)-[:Link\_to]->(l1),

(kd114:Commit {Id:'klo0369', Author:'Dmitry Kungurov', Message:'Fixed the remaining errors(ABEND S112, S10A)', Date:'2016-09-28'}),

(kd114)-[:Link\_to]->(l1),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd111)-[:Next]->(kd112),

(kd112)-[:Next]->(kd113),

(kd113)-[:Next]->(kd114),

(d1)-[:Committed\_the]->(kd111),

(d1)-[:Committed\_the]->(kd112),

(d1)-[:Committed\_the]->(kd113),

(d1)-[:Committed\_the]->(kd114),

//------------------------------------------------------------------------------

//Изменения / COMMITS D12 (Dmitry Kungurov) ------------------------------------

(kd121:Commit {Id:'fp49015', Author:'Dmitry Kungurov', Message:'Fixed the passing parameters', Date:'2016-09-30'}),

(kd121)-[:Link\_to]->(l1),

(kd122:Commit {Id:'fq5014o', Author:'Dmitry Kungurov', Message:'Using MACRO commands for call', Date:'2016-10-01'}),

(kd122)-[:Link\_to]->(l1),

(kd123:Commit {Id:'rty0115', Author:'Dmitry Kungurov', Message:'Fixed ABEND S013', Date:'2016-10-02'}),

(kd123)-[:Link\_to]->(l1),

(kd124:Commit {Id:'89543jq', Author:'Dmitry Kungurov', Message:'Added parsing for new parm', Date:'2016-10-02'}),

(kd124)-[:Link\_to]->(l1),

(kd125:Commit {Id:'tiv4581', Author:'Dmitry Kungurov', Message:'Fixed the open subsystem executor module', Date:'2016-10-04'}),

(kd125)-[:Link\_to]->(l1),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd121)-[:Next]->(kd122),

(kd122)-[:Next]->(kd123),

(kd123)-[:Next]->(kd124),

(kd124)-[:Next]->(kd125),

(d1)-[:Committed\_the]->(kd121),

(d1)-[:Committed\_the]->(kd122),

(d1)-[:Committed\_the]->(kd123),

(d1)-[:Committed\_the]->(kd124),

(d1)-[:Committed\_the]->(kd125),

//------------------------------------------------------------------------------

//Изменения / COMMITS D13 (Dmitry Kungurov) ------------------------------------

(kd131:Commit {Id:'5678vb1', Author:'Dmitry Kungurov', Message:'Retrieving parameters from C', Date:'2016-10-02'}),

(kd131)-[:Link\_to]->(l1),

(kd132:Commit {Id:'gh8912o', Author:'Dmitry Kungurov', Message:'Fixed ABEND S0C1', Date:'2016-10-03'}),

(kd132)-[:Link\_to]->(l1),

(kd133:Commit {Id:'4862fy8', Author:'Dmitry Kungurov', Message:'Added error and warning messages for all cases', Date:'2016-10-05'}),

(kd133)-[:Link\_to]->(l1),

(kd134:Commit {Id:'cv16726', Author:'Dmitry Kungurov', Message:'Fixed invalid function request', Date:'2016-10-06'}),

(kd134)-[:Link\_to]->(l1),

(kd135:Commit {Id:'45963hq', Author:'Dmitry Kungurov', Message:'Fixed parsing function', Date:'2016-10-08'}),

(kd135)-[:Link\_to]->(l1),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd131)-[:Next]->(kd132),

(kd132)-[:Next]->(kd133),

(kd133)-[:Next]->(kd134),

(kd134)-[:Next]->(kd135),

(d1)-[:Committed\_the]->(kd131),

(d1)-[:Committed\_the]->(kd132),

(d1)-[:Committed\_the]->(kd133),

(d1)-[:Committed\_the]->(kd134),

(d1)-[:Committed\_the]->(kd135),

//------------------------------------------------------------------------------

//Изменения / COMMITS D14 (Dmitry Kungurov) ------------------------------------

(kd141:Commit {Id:'rt70vy1', Author:'Dmitry Kungurov', Message:'Added the VERIFICATION FUNCTION for VSAM DS', Date:'2016-09-26'}),

(kd141)-[:Link\_to]->(l1),

(kd142:Commit {Id:'ey77901', Author:'Dmitry Kungurov', Message:'Fixed abnormal termination to avoid data integrity problems', Date:'2016-09-27'}),

(kd142)-[:Link\_to]->(l1),

(kd143:Commit {Id:'qr44781', Author:'Dmitry Kungurov', Message:'Resolved VVDS error', Date:'2016-09-28'}),

(kd143)-[:Link\_to]->(l1),

(kd144:Commit {Id:'nhe1570', Author:'Dmitry Kungurov', Message:'Fixed CSI', Date:'2016-09-29'}),

(kd144)-[:Link\_to]->(l1),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями------------------------------

(kd141)-[:Next]->(kd142),

(kd142)-[:Next]->(kd143),

(kd143)-[:Next]->(kd144),

(d1)-[:Committed\_the]->(kd141),

(d1)-[:Committed\_the]->(kd142),

(d1)-[:Committed\_the]->(kd143),

(d1)-[:Committed\_the]->(kd144),

//------------------------------------------------------------------------------

//Изменения / COMMITS D15 (Dmitry Kungurov) ------------------------------------

(kd151:Commit {Id:'oo3578b', Author:'Dmitry Kungurov', Message:'Description file was created and ABEND SCOC1 fixed', Date:'2016-09-19'}),

(kd151)-[:Link\_to]->(l1),

(kd152:Commit {Id:'09vbnio', Author:'Dmitry Kungurov', Message:'Update parameters in the description file', Date:'2016-09-23'}),

(kd152)-[:Link\_to]->(l1),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями------------------------------

(kd151)-[:Next]->(kd152),

(d1)-[:Committed\_the]->(kd151),

(d1)-[:Committed\_the]->(kd152),

//------------------------------------------------------------------------------

//Изменения / COMMITS D21 (Kamil Hasanov) --------------------------------------

(kd211:Commit {Id:'5863rk0', Author:'Kamil Hasanov', Message:'Added realization', Date:'2016-09-19'}),

(kd211)-[:Link\_to]->(l2),

(kd212:Commit {Id:'034bner', Author:'Kamil Hasanov', Message:'ABEND 013 fixed', Date:'2016-09-23'}),

(kd212)-[:Link\_to]->(l2),

(kd213:Commit {Id:'ty34356', Author:'Kamil Hasanov', Message:'The Dataset name field was expanded to 60 bytes', Date:'2016-09-25'}),

(kd213)-[:Link\_to]->(l2),

(kd214:Commit {Id:'tzb1357', Author:'Kamil Hasanov', Message:'Refactored linkage convension,using HLASM Macro', Date:'2016-09-26'}),

(kd214)-[:Link\_to]->(l2),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd211)-[:Next]->(kd212),

(kd212)-[:Next]->(kd213),

(kd213)-[:Next]->(kd214),

(d2)-[:Committed\_the]->(kd211),

(d2)-[:Committed\_the]->(kd212),

(d2)-[:Committed\_the]->(kd213),

(d2)-[:Committed\_the]->(kd214),

//------------------------------------------------------------------------------

//Изменения / COMMITS D22 (Kamil Hasanov) --------------------------------------

(kd221:Commit {Id:'ll456oh', Author:'Kamil Hasanov', Message:'Added realization', Date:'2016-09-24'}),

(kd221)-[:Link\_to]->(l2),

(kd222:Commit {Id:'uze0912', Author:'Kamil Hasanov', Message:'Fixed Abend with incorrect value parsing', Date:'2016-09-25'}),

(kd222)-[:Link\_to]->(l2),

(kd223:Commit {Id:'bbm5016', Author:'Kamil Hasanov', Message:'Fixed an Abend with not enough storage', Date:'2016-09-25'}),

(kd223)-[:Link\_to]->(l2),

(kd224:Commit {Id:'fk71935', Author:'Kamil Hasanov', Message:'Fixed ABEND S0C4', Date:'2016-09-27'}),

(kd224)-[:Link\_to]->(l2),

(kd225:Commit {Id:'gh257b0', Author:'Kamil Hasanov', Message:'Fixed ABEND S3C5', Date:'2016-09-29'}),

(kd225)-[:Link\_to]->(l2),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd221)-[:Next]->(kd222),

(kd222)-[:Next]->(kd223),

(kd223)-[:Next]->(kd224),

(kd224)-[:Next]->(kd225),

(d2)-[:Committed\_the]->(kd221),

(d2)-[:Committed\_the]->(kd222),

(d2)-[:Committed\_the]->(kd223),

(d2)-[:Committed\_the]->(kd224),

(d2)-[:Committed\_the]->(kd225),

//------------------------------------------------------------------------------

//Изменения / COMMITS D23 (Kamil Hasanov) --------------------------------------

(kd231:Commit {Id:'cc00124', Author:'Kamil Hasanov', Message:'Added realization for seek', Date:'2016-09-27'}),

(kd231)-[:Link\_to]->(l2),

(kd232:Commit {Id:'pj479xv', Author:'Kamil Hasanov', Message:'Fixed overflow exception S0CA', Date:'2016-09-28'}),

(kd232)-[:Link\_to]->(l2),

(kd233:Commit {Id:'34789ll', Author:'Kamil Hasanov', Message:'Fixed time service routine error', Date:'2016-09-29'}),

(kd233)-[:Link\_to]->(l2),

(kd234:Commit {Id:'0985mjy', Author:'Kamil Hasanov', Message:'Using Structed cycles', Date:'2016-09-29'}),

(kd234)-[:Link\_to]->(l2),

(kd235:Commit {Id:'flq7401', Author:'Kamil Hasanov', Message:'Fix invalid types specified', Date:'2016-09-30'}),

(kd235)-[:Link\_to]->(l2),

(kd236:Commit {Id:'nyr3915', Author:'Kamil Hasanov', Message:'Fixed Abend with incorrect value parsing', Date:'2016-10-01'}),

(kd236)-[:Link\_to]->(l2),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd231)-[:Next]->(kd232),

(kd232)-[:Next]->(kd233),

(kd233)-[:Next]->(kd234),

(kd234)-[:Next]->(kd235),

(kd235)-[:Next]->(kd236),

(d2)-[:Committed\_the]->(kd231),

(d2)-[:Committed\_the]->(kd232),

(d2)-[:Committed\_the]->(kd233),

(d2)-[:Committed\_the]->(kd234),

(d2)-[:Committed\_the]->(kd235),

(d2)-[:Committed\_the]->(kd236),

//------------------------------------------------------------------------------

//Изменения / COMMITS D24 (Kamil Hasanov) --------------------------------------

(kd241:Commit {Id:'65872hy', Author:'Kamil Hasanov', Message:'Coded simple read operation on DS', Date:'2016-10-02'}),

(kd241)-[:Link\_to]->(l2),

(kd242:Commit {Id:'fg56912', Author:'Kamil Hasanov', Message:'Added support read operation with big data', Date:'2016-10-03'}),

(kd242)-[:Link\_to]->(l2),

(kd243:Commit {Id:'91554by', Author:'Kamil Hasanov', Message:'Added multi-volume support', Date:'2016-10-05'}),

(kd243)-[:Link\_to]->(l2),

(kd244:Commit {Id:'78126uh', Author:'Kamil Hasanov', Message:'ABEND S002 fixed', Date:'2016-10-06'}),

(kd244)-[:Link\_to]->(l2),

(kd245:Commit {Id:'31975iz', Author:'Kamil Hasanov', Message:'SHOWDCB and MODCB macros was added before READ macro', Date:'2016-10-07'}),

(kd245)-[:Link\_to]->(l2),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd241)-[:Next]->(kd242),

(kd242)-[:Next]->(kd243),

(kd243)-[:Next]->(kd244),

(kd244)-[:Next]->(kd245),

(d2)-[:Committed\_the]->(kd241),

(d2)-[:Committed\_the]->(kd242),

(d2)-[:Committed\_the]->(kd243),

(d2)-[:Committed\_the]->(kd244),

(d2)-[:Committed\_the]->(kd245),

//------------------------------------------------------------------------------

//Изменения / COMMITS D25 (Kamil Hasanov) --------------------------------------

(kd251:Commit {Id:'zp1056m', Author:'Kamil Hasanov', Message:'Coded simple write operation on DS', Date:'2016-10-10'}),

(kd251)-[:Link\_to]->(l2),

(kd252:Commit {Id:'75345zo', Author:'Kamil Hasanov', Message:'ABEND SB36 fixed', Date:'2016-10-11'}),

(kd252)-[:Link\_to]->(l2),

(kd253:Commit {Id:'gh789as', Author:'Kamil Hasanov', Message:'Added IDCAMS support', Date:'2016-10-12'}),

(kd253)-[:Link\_to]->(l2),

(kd254:Commit {Id:'akk7315', Author:'Kamil Hasanov', Message:'Added checking on null buffer', Date:'2016-10-14'}),

(kd254)-[:Link\_to]->(l2),

(kd255:Commit {Id:'456nm78', Author:'Kamil Hasanov', Message:'Corrected cycle condition with Structured macros', Date:'2016-10-16'}),

(kd255)-[:Link\_to]->(l2),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd251)-[:Next]->(kd252),

(kd252)-[:Next]->(kd253),

(kd253)-[:Next]->(kd254),

(kd254)-[:Next]->(kd255),

(d2)-[:Committed\_the]->(kd251),

(d2)-[:Committed\_the]->(kd252),

(d2)-[:Committed\_the]->(kd253),

(d2)-[:Committed\_the]->(kd254),

(d2)-[:Committed\_the]->(kd255),

//------------------------------------------------------------------------------

//Изменения / COMMITS D31 (Vladislav Karakozov) --------------------------------

(kd311:Commit {Id:'mn75863', Author:'Vladislav Karakozov', Message:'Was coded a OPEN for KSDS using 24 AMODE and RMODE. Added open VSAM Multi-Volumes using this entry', Date:'2016-09-23'}),

(kd311)-[:Link\_to]->(l3),

(kd312:Commit {Id:'yy789z8', Author:'Vladislav Karakozov', Message:'Fixed an Abend S0C31 in case if we issue API on empty data set', Date:'2016-09-24'}),

(kd312)-[:Link\_to]->(l3),

(kd313:Commit {Id:'ttj3356', Author:'Vladislav Karakozov', Message:'The Dataset name field was expanded to 60 bytes', Date:'2016-09-26'}),

(kd313)-[:Link\_to]->(l3),

(kd314:Commit {Id:'ggu7705', Author:'Vladislav Karakozov', Message:'Added parsing for new parm', Date:'2016-09-27'}),

(kd314)-[:Link\_to]->(l3),

(kd315:Commit {Id:'hy3478z', Author:'Vladislav Karakozov', Message:'Refactored linkage convension', Date:'2016-09-28'}),

(kd315)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd311)-[:Next]->(kd312),

(kd312)-[:Next]->(kd313),

(kd313)-[:Next]->(kd314),

(kd314)-[:Next]->(kd315),

(d3)-[:Committed\_the]->(kd311),

(d3)-[:Committed\_the]->(kd312),

(d3)-[:Committed\_the]->(kd313),

(d3)-[:Committed\_the]->(kd314),

(d3)-[:Committed\_the]->(kd315),

//------------------------------------------------------------------------------

//Изменения / COMMITS D32 (Vladislav Karakozov) --------------------------------

(kd321:Commit {Id:'qs79131', Author:'Vladislav Karakozov', Message:'Added checking on null buffer', Date:'2016-09-28'}),

(kd321)-[:Link\_to]->(l3),

(kd322:Commit {Id:'aer6001', Author:'Vladislav Karakozov', Message:'Corrected cycle condition with Structured macros', Date:'2016-09-30'}),

(kd322)-[:Link\_to]->(l3),

(kd323:Commit {Id:'vp45091', Author:'Vladislav Karakozov', Message:'Added error and warning messages for all cases with extra API finalization', Date:'2016-10-01'}),

(kd323)-[:Link\_to]->(l3),

(kd324:Commit {Id:'q5914mn', Author:'Vladislav Karakozov', Message:'Fixed Abend with multi-volume DS.SHOWDCB and MODCB macros was added before READ macro', Date:'2016-10-02'}),

(kd324)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd321)-[:Next]->(kd322),

(kd322)-[:Next]->(kd323),

(kd323)-[:Next]->(kd324),

(d3)-[:Committed\_the]->(kd321),

(d3)-[:Committed\_the]->(kd322),

(d3)-[:Committed\_the]->(kd323),

(d3)-[:Committed\_the]->(kd324),

//------------------------------------------------------------------------------

//Изменения / COMMITS D33 (Vladislav Karakozov) --------------------------------

(kd331:Commit {Id:'fg45641', Author:'Vladislav Karakozov', Message:'Coded simple write operation on DS', Date:'2016-10-02'}),

(kd331)-[:Link\_to]->(l3),

(kd332:Commit {Id:'bng4891', Author:'Vladislav Karakozov', Message:'Using Structed cycles', Date:'2016-10-03'}),

(kd332)-[:Link\_to]->(l3),

(kd333:Commit {Id:'67091sl', Author:'Vladislav Karakozov', Message:'Added one more routine which could validate volume before write', Date:'2016-10-04'}),

(kd333)-[:Link\_to]->(l3),

(kd334:Commit {Id:'351klr1', Author:'Vladislav Karakozov', Message:'Added IDCAMS support', Date:'2016-10-05'}),

(kd334)-[:Link\_to]->(l3),

(kd335:Commit {Id:'vh16936', Author:'Vladislav Karakozov', Message:'Added support write operation with big data', Date:'2016-10-06'}),

(kd335)-[:Link\_to]->(l3),

(kd336:Commit {Id:'fgj3356', Author:'Vladislav Karakozov', Message:'Fixed a ABEND in case of not enough storage on Z/OS pool', Date:'2016-10-07'}),

(kd336)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd331)-[:Next]->(kd332),

(kd332)-[:Next]->(kd333),

(kd333)-[:Next]->(kd334),

(kd334)-[:Next]->(kd335),

(kd335)-[:Next]->(kd336),

(d3)-[:Committed\_the]->(kd331),

(d3)-[:Committed\_the]->(kd332),

(d3)-[:Committed\_the]->(kd333),

(d3)-[:Committed\_the]->(kd334),

(d3)-[:Committed\_the]->(kd335),

(d3)-[:Committed\_the]->(kd336),

//------------------------------------------------------------------------------

//Изменения / COMMITS D34 (Vladislav Karakozov) --------------------------------

(kd341:Commit {Id:'1245rk0', Author:'Vladislav Karakozov', Message:'Added realization', Date:'2016-09-19'}),

(kd341)-[:Link\_to]->(l3),

(kd342:Commit {Id:'4321gh3', Author:'Vladislav Karakozov', Message:'Calling REXX for main process of creation DS', Date:'2016-09-20'}),

(kd342)-[:Link\_to]->(l3),

(kd343:Commit {Id:'0078po1', Author:'Vladislav Karakozov', Message:'Added multi-volume support', Date:'2016-09-23'}),

(kd343)-[:Link\_to]->(l3),

(kd344:Commit {Id:'02vbn74', Author:'Vladislav Karakozov', Message:'Checking availability of volume before creation', Date:'2016-09-25'}),

(kd344)-[:Link\_to]->(l3),

(kd345:Commit {Id:'mm77326', Author:'Vladislav Karakozov', Message:'Added RC after REXX routine', Date:'2016-09-27'}),

(kd345)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd341)-[:Next]->(kd342),

(kd342)-[:Next]->(kd343),

(kd343)-[:Next]->(kd344),

(kd344)-[:Next]->(kd345),

(d3)-[:Committed\_the]->(kd341),

(d3)-[:Committed\_the]->(kd342),

(d3)-[:Committed\_the]->(kd343),

(d3)-[:Committed\_the]->(kd344),

(d3)-[:Committed\_the]->(kd345),

//------------------------------------------------------------------------------

//Изменения / COMMITS D35 (Vladislav Karakozov) --------------------------------

(kd351:Commit {Id:'19536rz', Author:'Vladislav Karakozov', Message:'Added realization', Date:'2016-10-08'}),

(kd351)-[:Link\_to]->(l3),

(kd352:Commit {Id:'1489zx1', Author:'Vladislav Karakozov', Message:'Fixed an ABEND with non-opened DS', Date:'2016-10-09'}),

(kd352)-[:Link\_to]->(l3),

(kd353:Commit {Id:'fgk1739', Author:'Vladislav Karakozov', Message:'DCB structure was corrected', Date:'2016-10-10'}),

(kd353)-[:Link\_to]->(l3),

(kd354:Commit {Id:'579jk78', Author:'Vladislav Karakozov', Message:'Fixed an ABEND with not enough storage', Date:'2016-10-13'}),

(kd354)-[:Link\_to]->(l3),

(kd355:Commit {Id:'fg0912j', Author:'Vladislav Karakozov', Message:'Added warning messages for non-existing DS', Date:'2016-10-14'}),

(kd355)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями-------------------------------

(kd351)-[:Next]->(kd352),

(kd352)-[:Next]->(kd353),

(kd353)-[:Next]->(kd354),

(kd354)-[:Next]->(kd355),

(d3)-[:Committed\_the]->(kd351),

(d3)-[:Committed\_the]->(kd352),

(d3)-[:Committed\_the]->(kd353),

(d3)-[:Committed\_the]->(kd354),

(d3)-[:Committed\_the]->(kd355),

//------------------------------------------------------------------------------

//Создание изменений мастера /COMMITS MASTER------------------------------------

(km1:Commit {Id:'77123bv', Author:'Roman Lobov', Message:'CREATE function for KSDS module was created', Date:'2016-09-19'}),

(km1)-[:Link\_to]->(l4),

(km2:Commit {Id:'771mn32', Author:'Roman Lobov', Message:'OPEN function for DA module was created', Date:'2016-09-19'}),

(km2)-[:Link\_to]->(l4),

(km3:Commit {Id:'99254cv', Author:'Roman Lobov', Message:'Creating descriptionfile was created', Date:'2016-09-19'}),

(km3)-[:Link\_to]->(l4),

(km4:Commit {Id:'56vb478', Author:'Roman Lobov', Message:'OPEN function for KSDS module was created', Date:'2016-09-20'}),

(km4)-[:Link\_to]->(l4),

(km5:Commit {Id:'63451ss', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-23', Tag:'Release v1.0'}),

(km5)-[:Link\_to]->(l4),

(km6:Commit {Id:'rr678ol', Author:'Roman Lobov', Message:'Defining non-VSAM data set types was created', Date:'2016-09-24'}),

(km6)-[:Link\_to]->(l4),

(km7:Commit {Id:'yi56120', Author:'Roman Lobov', Message:'CLOSE function for DA module was created', Date:'2016-09-24'}),

(km7)-[:Link\_to]->(l4),

(km8:Commit {Id:'99365qq', Author:'Roman Lobov', Message:'Defining VSAM data set types was created', Date:'2016-09-26'}),

(km8)-[:Link\_to]->(l4),

(km9:Commit {Id:'ppl258u', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-26', Tag:'Release v1.0'}),

(km9)-[:Link\_to]->(l4),

(km10:Commit {Id:'bn3394z', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-26'}),

(km10)-[:Link\_to]->(l4),

(km11:Commit {Id:'try5490', Author:'Roman Lobov', Message:'SEEK function for DA module was created', Date:'2016-09-26'}),

(km11)-[:Link\_to]->(l4),

(km12:Commit {Id:'ju00625', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-27', Tag:'Release v1.0'}),

(km12)-[:Link\_to]->(l4),

(km13:Commit {Id:'jkp0132', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-28', Tag:'Release v1.0'}),

(km13)-[:Link\_to]->(l4),

(km14:Commit {Id:'xdt5113', Author:'Roman Lobov', Message:'READ function for KSDS module was created', Date:'2016-09-26'}),

(km14)-[:Link\_to]->(l4),

(km15:Commit {Id:'ll365io', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-29', Tag:'Release v1.0'}),

(km15)-[:Link\_to]->(l4),

(km16:Commit {Id:'zv12509', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-29', Tag:'Release v1.0'}),

(km16)-[:Link\_to]->(l4),

(km17:Commit {Id:'sf14826', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-29'}),

(km17)-[:Link\_to]->(l4),

(km18:Commit {Id:'zm1143b', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-09-30', Tag:'Release v1.0'}),

(km18)-[:Link\_to]->(l4),

(km19:Commit {Id:'mqy7712', Author:'Roman Lobov', Message:'Call other modules was created', Date:'2016-09-30'}),

(km19)-[:Link\_to]->(l4),

(km20:Commit {Id:'mri3916', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-01', Tag:'Release v1.0'}),

(km20)-[:Link\_to]->(l4),

(km21:Commit {Id:'ko4478z', Author:'Roman Lobov', Message:'READ function for DA module was created', Date:'2016-10-02'}),

(km21)-[:Link\_to]->(l4),

(km22:Commit {Id:'vi3853a', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-02', Tag:'Release v1.0'}),

(km22)-[:Link\_to]->(l4),

(km23:Commit {Id:'fgk5692', Author:'Roman Lobov', Message:'WRITE function for KSDS module was created', Date:'2016-10-02'}),

(km23)-[:Link\_to]->(l4),

(km24:Commit {Id:'78443ni', Author:'Roman Lobov', Message:'Parsing parameters was created', Date:'2016-10-02'}),

(km24)-[:Link\_to]->(l4),

(km25:Commit {Id:'56621qm', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-04', Tag:'Release v1.0'}),

(km25)-[:Link\_to]->(l4),

(km26:Commit {Id:'ghj1590', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-05'}),

(km26)-[:Link\_to]->(l4),

(km27:Commit {Id:'hj11590', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-07', Tag:'Release v1.0'}),

(km27)-[:Link\_to]->(l4),

(km28:Commit {Id:'vo1776c', Author:'Roman Lobov', Message:'CLOSE function for KSDS module was created', Date:'2016-10-08'}),

(km28)-[:Link\_to]->(l4),

(km29:Commit {Id:'kc53327', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-08', Tag:'Release v1.0'}),

(km29)-[:Link\_to]->(l4),

(km30:Commit {Id:'61103yz', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-08', Tag:'Release v1.0'}),

(km30)-[:Link\_to]->(l4),

(km31:Commit {Id:'bi4509p', Author:'Roman Lobov', Message:'WRITE function for DA module was created', Date:'2016-10-09'}),

(km31)-[:Link\_to]->(l4),

(km32:Commit {Id:'zw17823', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-15', Tag:'Release v1.0'}),

(km32)-[:Link\_to]->(l4),

(km33:Commit {Id:'wwq5770', Author:'Roman Lobov', Message:'developer merged with the master', Date:'2016-10-17', Tag:'Release v1.0'}),

(km33)-[:Link\_to]->(l4),

//------------------------------------------------------------------------------

//Создание отношений между внесенными изменениями мастера-----------------------

(km1)-[:Next]->(km2),

(km2)-[:Next]->(km3),

(km3)-[:Next]->(km4),

(km4)-[:Next]->(km5),

(km5)-[:Next]->(km6),

(km6)-[:Next]->(km7),

(km7)-[:Next]->(km8),

(km8)-[:Next]->(km9),

(km9)-[:Next]->(km10),

(km10)-[:Next]->(km11),

(km11)-[:Next]->(km12),

(km12)-[:Next]->(km13),

(km13)-[:Next]->(km14),

(km14)-[:Next]->(km15),

(km15)-[:Next]->(km16),

(km16)-[:Next]->(km17),

(km17)-[:Next]->(km18),

(km18)-[:Next]->(km19),

(km19)-[:Next]->(km20),

(km20)-[:Next]->(km21),

(km21)-[:Next]->(km22),

(km22)-[:Next]->(km23),

(km23)-[:Next]->(km24),

(km24)-[:Next]->(km25),

(km25)-[:Next]->(km26),

(km26)-[:Next]->(km27),

(km27)-[:Next]->(km28),

(km28)-[:Next]->(km29),

(km29)-[:Next]->(km30),

(km30)-[:Next]->(km31),

(km31)-[:Next]->(km32),

(km32)-[:Next]->(km33),

(m)-[:Committed\_the]->(km1),

(m)-[:Committed\_the]->(km2),

(m)-[:Committed\_the]->(km3),

(m)-[:Committed\_the]->(km4),

(m)-[:Committed\_the]->(km5),

(m)-[:Committed\_the]->(km6),

(m)-[:Committed\_the]->(km7),

(m)-[:Committed\_the]->(km8),

(m)-[:Committed\_the]->(km9),

(m)-[:Committed\_the]->(km10),

(m)-[:Committed\_the]->(km11),

(m)-[:Committed\_the]->(km12),

(m)-[:Committed\_the]->(km13),

(m)-[:Committed\_the]->(km14),

(m)-[:Committed\_the]->(km15),

(m)-[:Committed\_the]->(km16),

(m)-[:Committed\_the]->(km17),

(m)-[:Committed\_the]->(km18),

(m)-[:Committed\_the]->(km19),

(m)-[:Committed\_the]->(km20),

(m)-[:Committed\_the]->(km21),

(m)-[:Committed\_the]->(km22),

(m)-[:Committed\_the]->(km23),

(m)-[:Committed\_the]->(km24),

(m)-[:Committed\_the]->(km25),

(m)-[:Committed\_the]->(km26),

(m)-[:Committed\_the]->(km27),

(m)-[:Committed\_the]->(km28),

(m)-[:Committed\_the]->(km29),

(m)-[:Committed\_the]->(km30),

(m)-[:Committed\_the]->(km31),

(m)-[:Committed\_the]->(km32),

(m)-[:Committed\_the]->(km33),

//------------------------------------------------------------------------------

//Создание отношений старта изменений / COMMITS---------------------------------

(d11)-[:Start\_of]->(km6),

(d12)-[:Start\_of]->(km19),

(d13)-[:Start\_of]->(km24),

(d14)-[:Start\_of]->(km8),

(d15)-[:Start\_of]->(km3),

(d21)-[:Start\_of]->(km2),

(d22)-[:Start\_of]->(km7),

(d23)-[:Start\_of]->(km11),

(d24)-[:Start\_of]->(km21),

(d25)-[:Start\_of]->(km31),

(d31)-[:Start\_of]->(km4),

(d32)-[:Start\_of]->(km14),

(d33)-[:Start\_of]->(km23),

(d34)-[:Start\_of]->(km1),

(d35)-[:Start\_of]->(km28),

//------------------------------------------------------------------------------

//Создание отношений NEXT и MERGE (слияние)-------------------------------------

(km1)-[:Next]->(kd341),

(km2)-[:Next]->(kd211),

(km3)-[:Next]->(kd151),

(km4)-[:Next]->(kd311),

(kd152)-[:Merge]->(km5),

(km6)-[:Next]->(kd111),

(km7)-[:Next]->(kd221),

(km8)-[:Next]->(kd141),

(kd313)-[:Merge]->(km10),

(kd214)-[:Merge]->(km9),

(km11)-[:Next]->(kd231),

(kd345)-[:Merge]->(km12),

(kd315)-[:Merge]->(km13),

(kd114)-[:Merge]->(km15),

(km14)-[:Next]->(kd321),

(kd225)-[:Merge]->(km16),

(kd233)-[:Merge]->(km17),

(kd144)-[:Merge]->(km18),

(km19)-[:Next]->(kd121),

(kd236)-[:Merge]->(km20),

(kd324)-[:Merge]->(km22),

(km21)-[:Next]->(kd241),

(km23)-[:Next]->(kd331),

(km24)-[:Next]->(kd131),

(kd125)-[:Merge]->(km25),

(kd333)-[:Merge]->(km26),

(kd336)-[:Merge]->(km27),

(kd245)-[:Merge]->(km29),

(kd135)-[:Merge]->(km30),

(km28)-[:Next]->(kd351),

(km31)-[:Next]->(kd251),

(kd355)-[:Merge]->(km32),

(kd255)-[:Merge]->(km33)

//------------------------------------------------------------------------------

//THE\_END-----------------------------------------------------------------------

**ПРИЛОЖЕНИЕ Б**

Скрипт для проекта “AUDIO\_PORTAL”.

Код скрипта:

CREATE

//Создание вершин, связанных с описанием проекта--------------------------------

(p1:Project {name:'Audio portal'}),

(p2:Customer {firstname: 'Ivan',lastname: 'Bulov'}),

(p3:Project\_type {type:'Web project'}),

(p4:Project\_code {code:'F001'}),

(p5:Project\_status {status:'Active'}),

//------------------------------------------------------------------------------

//Вершина MASTER----------------------------------------------------------------

(m:Master {firstname:'Dmitry',lastname:'Ivanov'}),

//------------------------------------------------------------------------------

//Вершины DEVELOPERS------------------------------------------------------------

(d1:Developer {firstname:'Oleg',lastname:'Sorokoin'}),

(d2:Developer {firstname:'Maxim',lastname:'Abramov'}),

//------------------------------------------------------------------------------

//Вершины ISSUES----------------------------------------------------------------

(i1:Issue {name:'Developing web interface'}),

(i2:Issue {name:'Register a domain'}),

//------------------------------------------------------------------------------

//Вершины описания ISSUES-------------------------------------------------------

(i21:Issue\_type {type:'Task'}),

(i22:Issue\_status {status:'Done'}),

(i23:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

(i11:Issue\_type {type:'New feature'}),

(i12:Issue\_status {status:'In progress'}),

(i13:Issue\_priority {priority:'High'}),

//------------------------------------------------------------------------------

//Создание отношений между вершинами / LINKS BETWEEN NODES----------------------

(p2)-[:Orders]->(p1),

(p1)-[:Type\_of\_project]->(p3),

(p1)-[:Code]->(p4),

(p1)-[:Status\_of\_project]->(p5),

(p1)-[:Consists\_of]->(i1),

(p1)-[:Consists\_of]->(i2),

//------------------------------------------------------------------------------

(m)-[:Defines]->(p3),

(m)-[:Assigns]->(p4),

(m)-[:Sets]->(p5),

(m)-[:Creates]->(p1),

(m)-[:Appoints]->(d1),

(m)-[:Appoints]->(d2),

(m)-[:Creates]->(i1),

(m)-[:Creates]->(i2),

(m)-[:Defines]->(i11),

(m)-[:Defines]->(i21),

(m)-[:Assigns]->(i13),

(m)-[:Assigns]->(i23),

//------------------------------------------------------------------------------

(d1)-[:Executes\_the]->(i1),

(d2)-[:Executes\_the]->(i2),

(d1)-[:Sets]->(i12),

(d2)-[:Sets]->(i22),

//------------------------------------------------------------------------------

(i1)-[:Priority]->(i13),

(i1)-[:Status\_of\_issue]->(i12),

(i1)-[:Type\_of\_issue]->(i11),

//------------------------------------------------------------------------------

(i2)-[:Priority]->(i23),

(i2)-[:Status\_of\_issue]->(i22),

(i2)-[:Type\_of\_issue]->(i21),

//------------------------------------------------------------------------------

//Создание изменений / COMMITS--------------------------------------------------

(k1:Commit {Id:'726884b', Author:'Dmitry Ivanov', Message:'Developing web interface was created', Date:'2017-01-10'}),

(k2:Commit {Id:'531f646', Author:'Oleg Sorokin', Message:'Create page layouts', Date:'2017-01-10'}),

(k3:Commit {Id:'fx0884b', Author:'Dmitry Ivanov', Message:'Register the domain was created', Date:'2017-01-11'}),

(k4:Commit {Id:'f2586c0', Author:'Oleg Sorokin', Message:'Fix bugs layouts', Date:'2017-01-11'}),

(k5:Commit {Id:'318564b', Author:'Maxim Abramov', Message:'Checked with the help of WHOIS service if the domain name is free', Date:'2017-01-11'}),

(k6:Commit {Id:'986fb73', Author:'Oleg Sorokin', Message:'Creating multimedia and FLASH-elements', Date:'2017-01-13'}),

(k7:Commit {Id:'u18664i', Author:'Maxim Abramov', Message:'Conclude an agreement on the provision of services', Date:'2017-01-13'}),

(k8:Commit {Id:'aa251d7', Author:'Oleg Sorokin', Message:'Testing and making adjustments', Date:'2017-01-15'}),

(k9:Commit {Id:'qq4004i', Author:'Maxim Abramov', Message:'Order and pay for services', Date:'2017-01-15'}),

(k10:Commit {Id:'247d75a', Author:'Dmitry Ivanov', Message:'developer merged with the master', Date:'2017-01-16'}),

(k11:Commit {Id:'4e42724', Author:'Maxim Abramov', Message:'Send request for domain registration', Date:'2017-01-16'}),

(k12:Commit {Id:'za781d7', Author:'Oleg Sorokin', Message:'Big update with using HTML,CSS and JavaScript', Date:'2017-01-18'}),

(k13:Commit {Id:'d25b206', Author:'Dmitry Ivanov', Message:'developer merged with the master', Date:'2017-01-18'}),

(k14:Commit {Id:'mb12224', Author:'Maxim Abramov', Message:'The domain is registered as MYWAV.ru', Date:'2017-01-19'}),

(k15:Commit {Id:'z25b879', Author:'Dmitry Ivanov', Message:'developer merged with the master', Date:'2017-01-19'}),

//------------------------------------------------------------------------------

//Создание отношений старта изменений / COMMITS---------------------------------

(i1)-[:Start\_of]->(k1),

(i2)-[:Start\_of]->(k3),

//------------------------------------------------------------------------------

//Создание отношений от вершины MASTER------------------------------------------

(m)-[:Committed\_the]->(k1),

(m)-[:Committed\_the]->(k3),

(m)-[:Committed\_the]->(k10),

(m)-[:Committed\_the]->(k13),

(m)-[:Committed\_the]->(k15),

//------------------------------------------------------------------------------

//Создание отношений от вершины FIRST\_DEVELOPER (Developer)---------------------------------

(d1)-[:Committed\_the]->(k2),

(d1)-[:Committed\_the]->(k4),

(d1)-[:Committed\_the]->(k6),

(d1)-[:Committed\_the]->(k8),

(d1)-[:Committed\_the]->(k12),

//------------------------------------------------------------------------------

//Создание отношений от вершины SECOND\_DEVELOPER (Developer)--------------------

(d2)-[:Committed\_the]->(k5),

(d2)-[:Committed\_the]->(k7),

(d2)-[:Committed\_the]->(k9),

(d2)-[:Committed\_the]->(k11),

(d2)-[:Committed\_the]->(k14),

//------------------------------------------------------------------------------

//Создание вершин REPOSITORY----------------------------------------------------

(l1:Repository {name:'Github', URL:'https://github.com/Ivanov/audio\_portal'}),

(l2:Repository {name:'Bitbucket',URL:'https://bitbucket.org/Sorokin/AUDIOportal'}),

(l3:Repository {name:'Bitbucket',[URL:'https://bitbucket.org/Abramov/AUDIOportal'}](URL:'https://bitbucket.org/Abramov/AUDIOportal'%7d)),

//------------------------------------------------------------------------------

//Создание отношений ссылок / LINK----------------------------------------------

(i1)-[:Link\_to]->(l1),

(i2)-[:Link\_to]->(l1),

(k1)-[:Link\_to]->(l1),

(k3)-[:Link\_to]->(l1),

(k10)-[:Link\_to]->(l1),

(k13)-[:Link\_to]->(l1),

(k15)-[:Link\_to]->(l1),

(k2)-[:Link\_to]->(l2),

(k4)-[:Link\_to]->(l2),

(k6)-[:Link\_to]->(l2),

(k8)-[:Link\_to]->(l2),

(k12)-[:Link\_to]->(l2),

(k5)-[:Link\_to]->(l3),

(k7)-[:Link\_to]->(l3),

(k9)-[:Link\_to]->(l3),

(k11)-[:Link\_to]->(l3),

(k14)-[:Link\_to]->(l3),

//------------------------------------------------------------------------------

//Создание отношений NEXT и MERGE (слияние)-------------------------------------

(k1)-[:Next]->(k2),

(k1)-[:Next]->(k3),

(k2)-[:Next]->(k4),

(k3)-[:Next]->(k5),

(k4)-[:Next]->(k6),

(k6)-[:Next]->(k8),

(k8)-[:Next]->(k12),

(k8)-[:Merge]->(k10),

(k12)-[:Merge]->(k13),

(k3)-[:Next]->(k10),

(k10)-[:Next]->(k13),

(k13)-[:Next]->(k15),

(k5)-[:Next]->(k7),

(k7)-[:Next]->(k9),

(k9)-[:Next]->(k11),

(k11)-[:Next]->(k14),

(k14)-[:Merge]->(k15)

//------------------------------------------------------------------------------

//THE\_END