Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью ВКР является – Разработка базы данных хранилище файлов.

Актуальность заключается в том, что в наши дни файлов стало очень много и их нужно где-то хранить и сортировать, чем и является база данных «Хранилище файлов».

Объектом исследования является – хранение информации о файлах, папках и пользователях в реляционных базах данных.

Предметом исследования является – разработка базы данных для хранения файлов.

Результатом окончания ВКР будет – прототип разработанной базы данных.

Какие задачи включает в себя работа над ВКР:

1) Анализ предметной области;

2) Анализ существующих базы данных;

3) Анализ схемы базы данных;

4) Проектирование базы данных;

5) Разработка базы данных;

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исследование предметной области

1.1.1 Анализ основных понятий

Опираясь на выбранную тему, основными понятиями будут являться:

Файл – Специально организованная структура данных, распознаваемая компьютером как единое целое.

Хранилище файлов – база данных, хранящая в себе файлы.

Пользователь – лицо или группа лиц, которое использует действующую систему для выполнения конкретной функции.

1.1.2 Основные объекты

Основным объектом является:

Файл (таблица 1).

Таблица 1 – Сведения о файле.

Характеристики объекта	**Описание характеристик**
Тип	– какой тип файла.
Размер	– размер файла.
Название	– как называется файл.
Дополнительные сведения	– информация о создании, изменении, расположении файла и т.п.

1.1.3 Основные действия с объектами

Каждый из объектов имеет несколько основных действий:

1) Объект «файл»:

– открыть;

– закрыть;

2) Объект «пользователь»:

– отправить файл;

– переименовать;

– удалить;

1.1.4 Основные участники предметной области

Основным участником предметной области является пользователь. Это человек, который взаимодействует с файлами. Он может добавлять, удалять и управлять файлами в системе.

1.2 Анализ существующих СУБД

Neo4j

Анализ

Neo4j – графовая система управления базами данных с открытым исходным кодом, реализованная на Java. Данные хранит в собственном формате, специализированно приспособленном для представления графовой информации, такой подход в сравнении с моделированием графовой базы данных средствами реляционной СУБД позволяет применять дополнительную оптимизацию в случае данных с более сложной структурой. Также утверждается о наличии специальных оптимизаций для SSD-накопителей, при этом для обработки графа не требуется его помещение целиком в оперативную память вычислительного узла, таким образом, возможна обработка достаточно больших графов.

Основные возможности

Данная СУБД имеет следующие возможности:

- Гибкая модель данных – Neo4j предоставляет гибкую простую и вместе с тем мощную модель данных, которую можно легко изменять в зависимости от приложений и отраслей;
- Анализ в реальном времени – Neo4j предоставляет результаты на основе данных в реальном времени;
- Высокая доступность – No4j отлично доступен для крупных корпоративных приложений реального времени с транзакционными гарантиями;
- Простой поиск – Используя Neo4j, вы можете не только представлять, но и легко извлекать (перемещаться / перемещаться) связанные данные быстрее по сравнению с другими базами данных;

- Нет объединений – используя Neo4j, он НЕ требует сложных объединений для извлечения связанных / связанных данных, так как очень легко получить сведения о соседнем узле или взаимосвязи без объединений или индексов.

Типы данных

Типы недвижимости:

- Number, абстрактный тип, который имеет подтипы *Integer* и *Float;*
- Нить;
- Логический;
- Пространственный тип Point;
- Временные типы: Date, Time, LocalTime, DateTime, LocalDateTime и Duration.
- Структурные типы:
- Узлы (идентификатор, этикетка (и), карта (собственности));
- Отношения (идентификатор, тип, карта (собственности), идентификатор начального и конечного узлов);
- Пути (чередующаяся последовательность узлов и отношений).
- Составные типы:
- Списки – представляют собой разнородные упорядоченные наборы значений, каждое из которых имеет какое-либо свойство, структурный или составной тип.
- Карты – представляют собой разнородные неупорядоченные коллекции пар (ключ, значение).

Язык запроса

В СУБД используется собственный язык запросов – Cypher, но запросы можно делать и другими способами, например, напрямую через Java API и на языке Gremlin, созданном в проекте с открытым исходным кодом TinkerPop. Cypher является не только языком запросов, но и языком манипулирования данными, так как предоставляет функции CRUD для графового хранилища.

Cassandra

Анализ

Apache Cassandra – это не реляционная отказоустойчивая распределенная СУБД, рассчитанная на создание высоко масштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша. Проект был разработан на языке Java в корпорации Facebook в 2008 году, и передан фонду Apache Software Foundation в 2009. Эта СУБД относится к гибридным NoSQL-решениям, поскольку она сочетает модель хранения данных на базе семейства столбцов (ColumnFamily) с концепцией key-value (ключ-значение).

Основные возможности

Данная СУБД имеет следующие возможности:

1) Эластичная масштабируемость – Cassandra отлично масштабируется; это позволяет добавить больше оборудования, чтобы разместить больше клиентов и больше данных согласно требованию;

2) Всегда на архитектуре – Cassandra не имеет единой точки отказа, и она постоянно доступна для критически важных для бизнеса приложений, которые не могут допустить сбоя;

3) Высокая производительность в линейном масштабе – Cassandra линейно масштабируется, т. Е. Увеличивает пропускную способность при увеличении количества узлов в кластере. Поэтому он поддерживает быстрое время отклика;

4) Простое распространение данных – Cassandra обеспечивает гибкость в распределении данных там, где вам нужно, путем репликации данных между несколькими центрами обработки данных;

5) Поддержка транзакций – Cassandra поддерживает такие свойства, как атомарность, согласованность, изоляция и долговечность (ACID).

Типы данных

1) BytesType: любые байтовые строки (без валидации);

2) BytesType: любые байтовые строки (без валидации);

3) UTF8Type: UTF-8 строка;

4) IntegerType: число с произвольным размером;

5) Int32Type: 4-байтовое число;

6) LongType: 8-байтовое число;

7) TimeUUIDType: UUID 1-ого типа;

8) BooleanType: два значения: true = 1 или false = 0;

9) DoubleType: 8-байтовое число с плавающей запятой;

10) FloatType: 4-байтовое число с плавающей запятой;

11) DecimalType: число с произвольным размером и плавающей запятой.

Языки запроса

Пользователи могут получить доступ к Cassandra через его узлы, используя Cassandra Query Language (CQL). CQL рассматривает базу данных **(Keyspace)** как контейнер таблиц. Программисты используют **cqlsh:** приглашение работать с CQL или отдельными драйверами языка приложения.

Клиенты обращаются к любому из узлов за своими операциями чтения-записи. Этот узел (координатор) воспроизводит прокси между клиентом и узлами, содержащими данные.

MySQL

Анализ

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Основные возможности:

Полностью многопоточное использование ядерных нитей. Это означает, что пакет может легко использовать много CPUs, если они есть;

1) Интерфейсы для языков C, C++, Eiffel, Java, Perl, PHP, Python и Tcl;

2) Работает на многих различных платформах;

3) Очень быстрые объединения, использующие оптимизированное однопроходное объединение многих таблиц;

4) Полная поддержка операторов и функций в частях запроса SELECT и WHERE;

5) Поддержка LEFT OUTER JOIN и RIGHT OUTER JOIN с синтаксисами ANSI SQL и ODBC;

6) Привилегии и система паролей, которая является очень гибкой и безопасной, и позволяет проверку, основанную на имени хоста.

Типы данных

1) CHAR: представляет строку фиксированной длины;

2) VARCHAR: представляет строку переменной длины;

3) TINYTEXT: представляет текст длиной до 255 байт;

4) INT: представляет целые числа от -2147483648 до 2147483647, занимает 4 байта;

5) DECIMAL: хранит числа с фиксированной точностью. Данный тип может принимать два параметра precision и scale: DECIMAL (precision, scale);

6) FLOAT: хранит дробные числа с плавающей точкой одинарной точности;

7) DOUBLE: хранит дробные числа с плавающей точкой двойной точности.

Язык запроса

Общая структура запросов в MySQL выглядит следующим образом.

SELECT ('столбцы или * для выбора всех столбцов; обязательно');

FROM ('таблица; обязательно');

WHERE ('условие/фильтрация, например, city = 'Moscow'; необязательно');

GROUP BY ('столбец, по которому хотим сгруппировать данные; необязательно');

HAVING ('условие/фильтрация на уровне сгруппированных данных; необязательно');

ORDER BY ('столбец, по которому хотим отсортировать вывод; необязательно').

1.3 Обоснование выбора СУБД

Практически все разработчики современных приложений, предусматривающих связь с системами баз данных, ориентируются на реляционные СУБД. Основными являются Oracle, MySQL, MS SQL. Рассмотрим особенности этих СУБД:

1) Oracle обрабатывает большие данные, поддерживает SQL, Oracle NoSQL Database с Java/C API для чтения и записи данных.

2) MySQL имеет масштабируемость, скорость, безопасность, легка в использовании, поддержка многих операционных систем, поддерживает Novell Cluster.

3) MS SQL имеет высокую производительность, возможность устанавливать разные версии на одном компьютере, генерацию скриптов для перемещения данных.

Исходя из особенностей вышеперечисленных СУБД MySQL будет соответствовать требованиям для дальнейшей работы с предметной областью.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Формирование требований

2.1.1 Основные сущности базы данных

В БД будут храниться следующие сущности:

- Администратор - исправление неисправностей в системе, отслеживание запрещенных файлов.
- Пользователь - создание, просмотр и изменение файлов.
- Файл - основная единица в БД, с которой работают пользователи.
- Папка – создается пользователем для структурирования данных в системе.

Таблица 2 – Сущности базы данных

Сущность	**Характеристики**
Администратор	Логин Пароль Уровень доступа
Пользователь	Логин Пароль Уровень доступа
Файл	Тип Размер Название Дата создания Дата изменения Доступность
Папка	Размер Название Дата создания Дата изменения Доступность

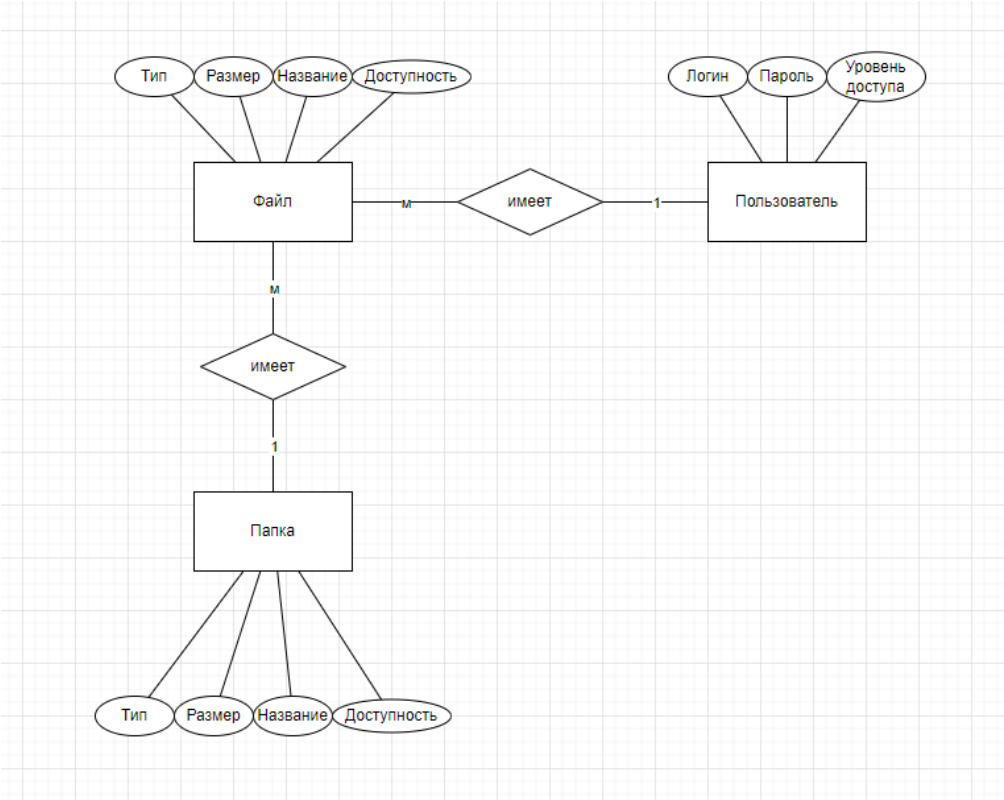

Рисунок 1 – Диаграмма сущностей

2.1.2 Варианты использования

Для разработки сценариев использования будет использоваться UML диаграмма использования, иначе use-cases. Она составляется для того, чтобы наглядно представить функциональные возможности разрабатываемой системы.

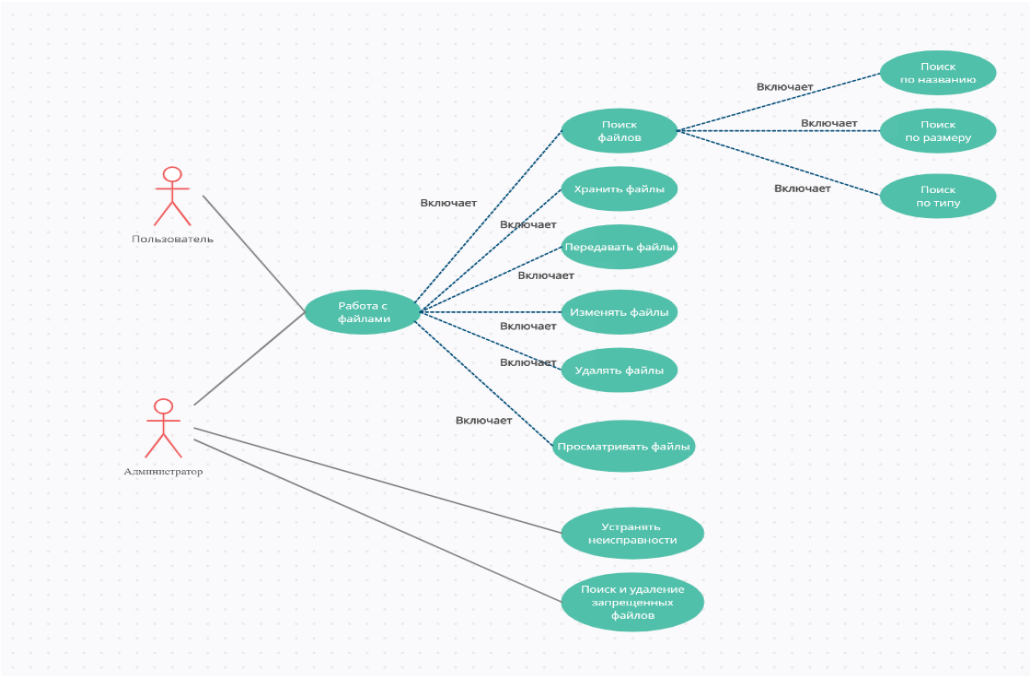

Рисунок 2 – use-case диаграмма

2.1.3 Определение API для взаимодействия с базой данных

Таблица 3 - API для взаимодействия с базой данных

Метод	Аргументы
Изменить файл	Тип Размер Название Дата создания Дата изменения Пользователь
Изменить папку	Тип Размер Название Дата создания Дата изменения Пользователь
Изменить название файла	Id файла Название файла Пользователь
Изменить тип файла	Id файла Тип файла Пользователь
Изменить название папки	Id файла Название папки Пользователь
Изменить тип папки	Id файла Тип папки Пользователь

2.2 Проектирование базы данных

2.2.1 Физическая схема базы данных

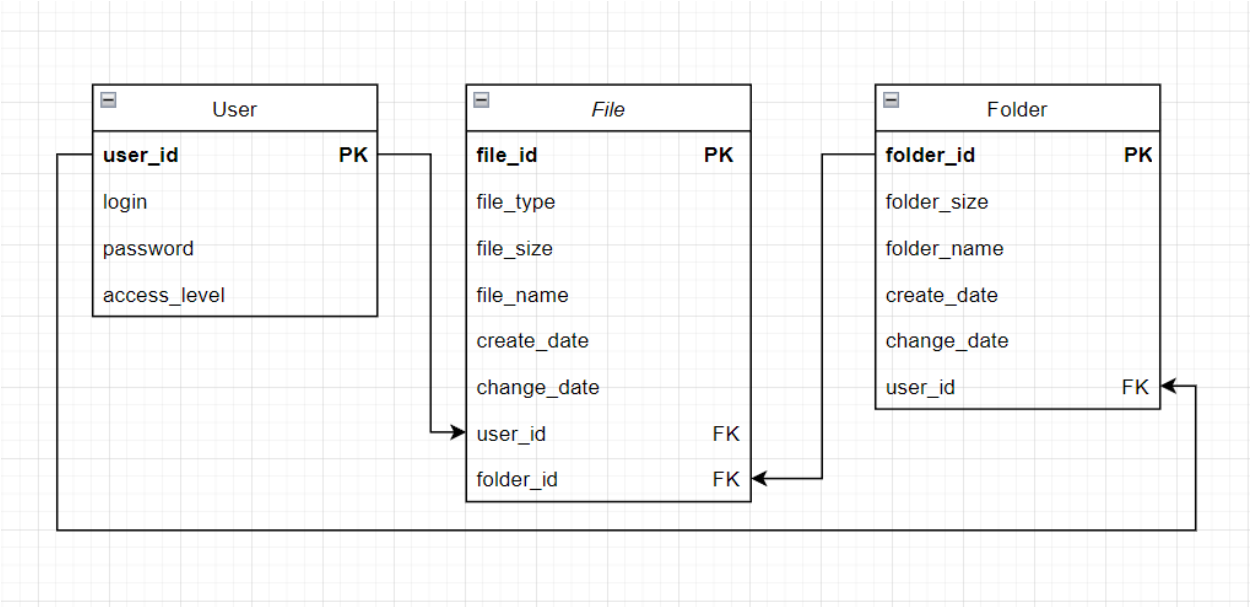

Рисунок 3 – физическая схема базы данных

2.2.3 Определение типовых запросов к объектам базы данных

Таблица 4 – Типовые запросы

Таблица	**Запрос**	**Пример**
Пользователи	Список пользователей	SELECT * FROM ‘User’
	Добавить пользователя	INSERT INTO ‘User’ (‘login’, ‘password’, ‘access_level’) VALUES (‘qwerty123’, ‘123321’, ‘stduser’)
	Изменить пользователя	UPDATE ‘User’ SET login = IsNull(@login, login), password= IsNull(@password, 43212321) WHERE user_id=1
	Удалить пользователя	DELETE FROM ‘User’ WHERE user_id=1
Файлы	Список файлов	SELECT * FROM ‘File’

Продолжение таблицы 4

Таблица	**Запрос**	**Пример**
	Добавить файл	INSERT INTO ‘File’ (‘file_id’, ‘file_type’, ‘file_size’, ‘file_name’, ‘create_date’, ‘user_id’) VALUES (‘1’, ‘jpg’, ‘24mb’, ‘photo’, ’10.10.2021’, ‘1’)
	Изменить файл	UPDATE ‘File’ SET file_name= IsNull(@file_name, photo1), user_id= IsNull(@user_id, 2) WHERE file_id=1
	Удалить файл	DELETE FROM ‘File’ WHERE file_id=1
Папки	Список папок	SELECT * FROM ‘Folder’
	Добавить папку	INSERT INTO ‘Folder’ (‘folder_id’, ‘folder_type’, ‘folder_size’, ‘folder_name’, ‘create_date’, ‘user_id’) VALUES (‘1’, ‘ ’, ‘1280mb’, ‘qwerty’, ’10.10.2021’, ‘1’)
	Изменить папку	UPDATE ‘Folder’ SET folder_name= IsNull(@folder_name, ytrewq), user_id= IsNull(@user_id, 2) WHERE folder_id=1
	Удалить папку	DELETE FROM ‘Folder’ WHERE folder_id=1

2.2.4 Определение процедур и функций API

Таблица 5 - процедуры и функции API

API функции	**Описание API функции**
changeFile(file_type (varchar50), file_size (varchar50), file_name (varchar50), create_date(date), change_date (date), user_id(varchar50))	Изменяет все данные файла

Продолжение таблицы 5

API функции	**Описание API функции**
changeFolder(folder_type (varchar50), folder_size (varchar50), folder_name (varchar50), create_date(date), change_date (date), user_id(varchar50))	Изменяет все данные папки
changeFileName(file_id(int), file_name (varchar50), user_id(varchar50))	Изменить название определенного файла
changeFileType(file_id(int), file_type (varchar50), user_id(varchar50))	Изменить тип определенного файла
changeFolderName(folder_id(int), folder_name (varchar50), user_id(varchar50))	Изменить название определенной папки
changeFolderType(folder_id(int), folder_type (varchar50), user_id(varchar50))	Изменить тип определенной папки

2.3 Разработка базы данных

2.3.1 Разработанные таблицы

Таблицы созданы в соответствии с физической схемой базы данных, код для создания данных таблиц представлен в Приложении

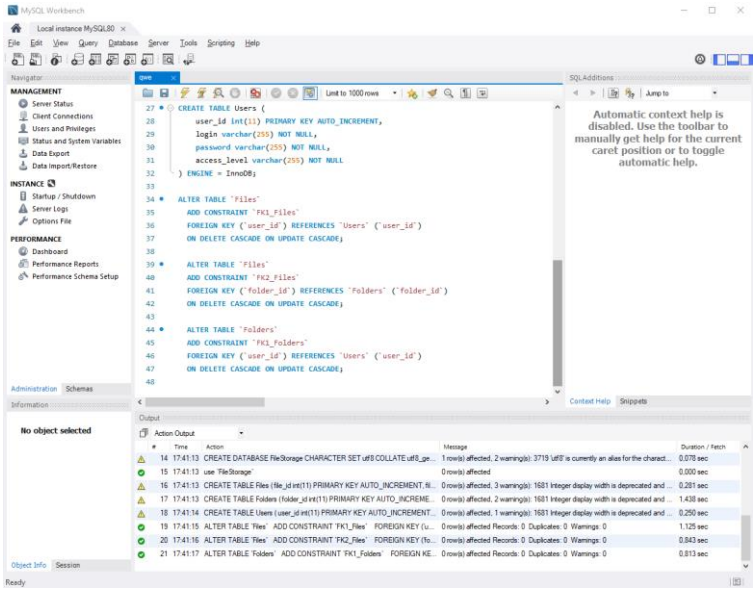

Рисунок 4 – Результат выполнения запроса на создание базы данных

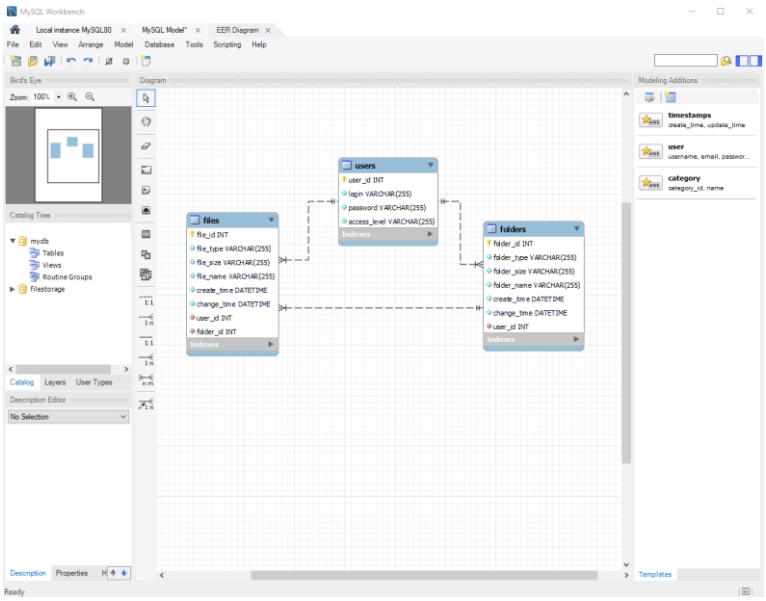

Рисунок 5 – Связи в таблице после ее создания через запрос

2.3.2 Разработанные представления

Описанные в требованиях представления разработаны и их исходный код описан в Приложении

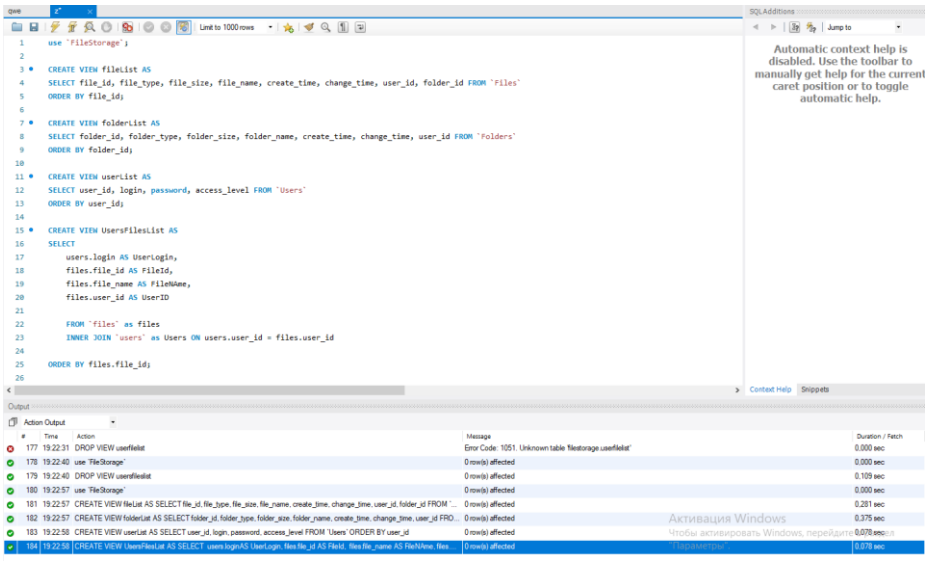

Рисунок 6 - Результат выполнения запроса на создание представлений

Рисунок 7 – Представления на диаграмме

2.3.3 Разработанные процедуры и функции

В соответствии с обозначенными требованиями были разработаны процедуры и функции, представленные в Приложении

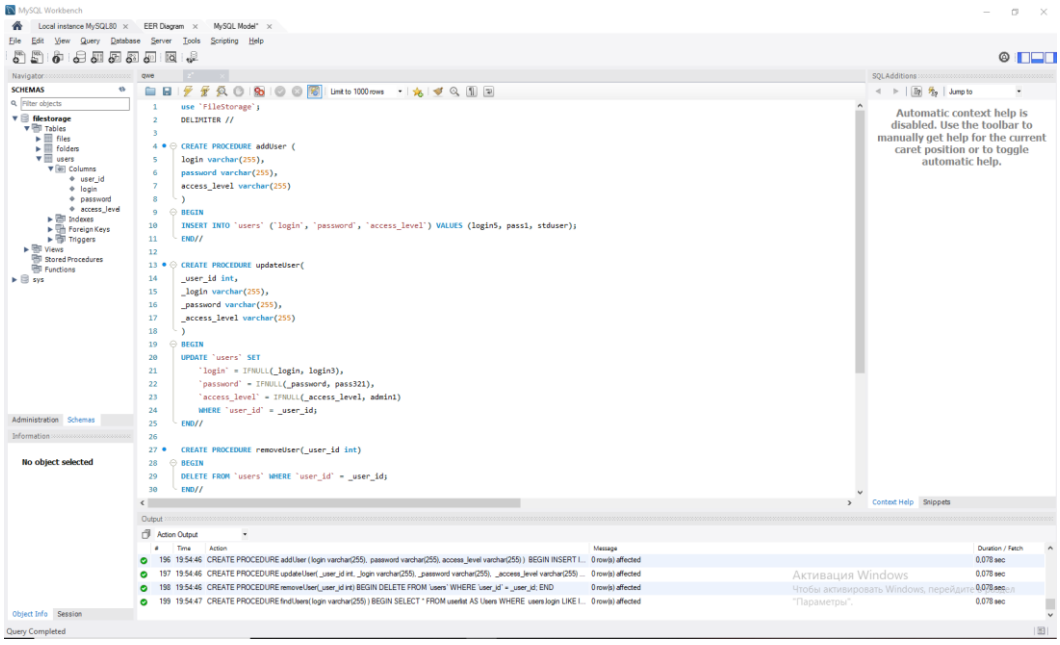

Рисунок 8 - Результат выполнения запроса на создание процедур.

3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР была разработана база данных хранение файлов.

В процессе выполнения ВКР были выполнены следующие задачи:

1) Анализ предметной области;

2) Анализ существующих базы данных;

3) Анализ схемы базы данных;

4) Проектирование базы данных;

5) Разработка базы данных.

Достоинства разработанной базы данных:

1) Возможность использования в небольших сервисах;

2) Расширяемость за счет возможностей выбранной СУБД;

3) Минимальные требования к знаниям для использования БД при использовании разработанного API;

4) Методы позволяют тратить меньше времени на добавление, изменение и удаление данных.

Недостатки разработанной базы данных:

1) Доступ пользователей на уровне БД не разграничен;

2) Отсутствие вспомогательных функций.

СПИСОК ЛИТЕРАТУРЫ

1) Официальный сайт MongoDB. [Электронный ресурс]. — URL: https://docs.mongodb.com/manual (Дата обращения: 29.11.2021).

2) Redis для начинающих. [Электронный ресурс]. — URL: https://webdevblog.ru/redis-dlya-nachinajushhij/ (Дата обращения: 29.11.2021).

3) MySQL Википедия. [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/MySQL (Дата обращения: 29.11.2021).

4) MongoDB Datatypes. [Электронный ресурс]. — URL: https://www.tutorialspoint.com/mongodb/mongodb_datatype.htm (Дата обращения: 29.11.2021).

5) Официальный сайт Redis. [Электронный ресурс]. — URL: https://redis.io/documentation (Дата обращения: 29.11.2021).

6) Основные команды SQL, которые должен знать каждый программист. [Электронный ресурс]. — URL: https://tproger.ru/translations/sql-recap/ (Дата обращения: 29.11.2021).

7) MySQL. [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/MySQL (Дата обращения: 23.11.2021).

8) Application Interface. [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/API (Дата обращения: 23.11.2021).

9) Основные команды SQL. [Электронный ресурс]. — URL: https://tproger.ru/translations/sql-recap/ (Дата обращения: 25.11.2021).

10) Оператор SQL PRIMARY KEY. [Электронный ресурс]. — URL: http://2sql.ru/novosti/sql-primary-key/ (Дата обращения: 25.11.2021).

11) Представления. [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15 (Дата обращения: 25.11.2021).

ПРИЛОЖЕНИЕ. ИСХОДНЫЙ КОД ОБЪЕКТОВ БАЗЫ ДАННЫХ

Таблица 6 - Исходный код создания БД и таблиц

Исходный код создания БД и таблиц

```
CREATE DATABASE FileStorage CHARACTER SET utf8 COLLATE utf8_general_ci;

use `FileStorage`;

CREATE TABLE Files (
        file_id int(11) PRIMARY KEY AUTO_INCREMENT,
        file_type varchar(255) NOT NULL,
        file_size varchar(255) NOT NULL,
        file_name varchar(255) NOT NULL,
        create_time datetime NOT NULL,
        change_time datetime NOT NULL,
   user_id int(11) NOT NULL,
   folder_id int(11) NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Folders (
        folder_id int(11) PRIMARY KEY AUTO_INCREMENT,
        folder_type varchar(255) NOT NULL,
        folder_size varchar(255) NOT NULL,
        folder_name varchar(255) NOT NULL,
        create_time datetime NOT NULL,
        change_time datetime NOT NULL,
   user_id int(11) NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Users (
        user_id int(11) PRIMARY KEY AUTO_INCREMENT,
        login varchar(255) NOT NULL,
        password varchar(255) NOT NULL,
        access_level varchar(255) NOT NULL
) ENGINE = InnoDB;

ALTER TABLE `Files`
 ADD CONSTRAINT `FK1_Files`
 FOREIGN KEY (`user_id`) REFERENCES `Users` (`user_id`)
 ON DELETE CASCADE ON UPDATE CASCADE;

 ALTER TABLE `Files`
 ADD CONSTRAINT `FK2_Files`
 FOREIGN KEY (`folder_id`) REFERENCES `Folders` (`folder_id`)
 ON DELETE CASCADE ON UPDATE CASCADE;

 ALTER TABLE `Folders`
 ADD CONSTRAINT `FK1_Folders`
 FOREIGN KEY (`user_id`) REFERENCES `Users` (`user_id`)
 ON DELETE CASCADE ON UPDATE CASCADE;
```

Таблица 7 - Исходный код создания хранимых процедур

Исходный код создания хранимых процедур

```
use `FileStorage`;

DELIMITER //

CREATE PROCEDURE addUser (
login varchar(255),
password varchar(255),
access_level varchar(255)
)
BEGIN
INSERT INTO `users` (`login`, `password`, `access_level`) VALUES (login5, pass1, stduser);
END//

CREATE PROCEDURE updateUser(
_user_id int,
_login varchar(255),
_password varchar(255),
_access_level varchar(255)
)
BEGIN
UPDATE `users` SET
        `login` = IFNULL(_login, login3),
   `password` = IFNULL(_password, pass321),
   `access_level` = IFNULL(_access_level, admin1)
   WHERE `user_id` = _user_id;
END//

CREATE PROCEDURE removeUser(_user_id int)
BEGIN
DELETE FROM `users` WHERE `user_id` = _user_id;
END//

CREATE PROCEDURE findUsers(
login varchar(255)
)
BEGIN
SELECT * FROM userlist AS Users
WHERE
        users.login LIKE IFNULL(login, '%');
END//
```

Таблица 8 - Исходный код создания представлений

Исходный код создания представлений

```
use `FileStorage`;

CREATE VIEW fileList AS
SELECT file_id, file_type, file_size, file_name, create_time, change_time, user_id, folder_id
FROM `Files`
ORDER BY file_id;

CREATE VIEW folderList AS
SELECT folder_id, folder_type, folder_size, folder_name, create_time, change_time, user_id FROM
`Folders`
ORDER BY folder_id;

CREATE VIEW userList AS
SELECT user id, login, password, access level FROM `Users`
ORDER BY user id;

CREATE VIEW UsersFilesList AS
SELECT
          users.login        AS UserLogin,
          files.file id AS FileId,
          files.file_name AS FileNAme,
          files.user id AS UserID

    FROM `files` as files
          INNER JOIN `users` as Users ON users.user_id = files.user_id

ORDER BY files.file id;
```