Слайд 1:

Здравствуйте, тема моей работы - система автоматического сбора информации о работе NoSQL баз данных.

Слайд 2:

Целью является создание системы автоматического сбора информации о работе NoSQL баз данных, которая позволяет сделать процесс получения и обработки информации из разных БД более удобным для пользователей.

Для этого необходимы выполнить следующие задачи:

Произвести анализ средств извлечения информации из Nosql СУБД;

Исследовать методы и технологии извлечения информации из нескольких СУБД;

Разработать структуру и архитектуру системы;

Разработать грамматику языка для запросов к системе;

Создать метод и алгоритм разбиения запроса к нескольким СУБД на подзапросы для каждой;

Разработать макет интерфейса системы;

Произвести тестирование разработанной системы.

Слайд 3:

В самой системы предусмотрены следующие функциональные возможности:

Взаимодействие с данными (добавление, удаления, обновление, чтение) из разных БД с помощью созданного языка запросов;

Взаимодействие с индексами, а именно создание, удаление, просмотр созданных;

Получение списка всех таблиц из БД;

Показ структуры выбранной таблицы;

Показ количества записей в выбранной таблице;

Показ времени выполнения запроса и загрузки оперативной памяти при исполнении запроса;

Показ плана запроса;

Запуск запроса на языке СУБД, для которой он предназначается.

Слайд 4:

Для выполнения функциональных требований был произведён анализ средств извлечения информации для каждой СУБД. На данном слайде представлены команды, которые используются в системе для получения информации из MongoDB, которая является документоориентированная СУБД.

Слайд 5:

На данном слайде представлены команды для получения информации из Neo4j, которая является графовой СУБД.

Слайд 6:

На данном слайде представлены команды для получения информации из Cassandra, которая является колоночной СУБД.

Слайд 7:

Какие бывают методы извлечения информации из нескольких СУБД.

Узконаправленные системы.

В данной группе систем специально создаются системы, которые включают в себя заранее определённые БД, как правило связанные общей тематикой.

Ручное объединение БД.

Данная группа пытается вручную объединить несколько разных СУБД. Для этого имеется несколько подходов.

Первый подход. Получить все имеющиеся схемы БД и, сравнивая и редактируя их, можно добиться слияния в единую схему БД.

Второй подход – разработка интегрированной схемы.

Слайд 8:

Создание новой СУБД.

Для решения задачи связи нескольких СУБД можно разработать совершенно новые СУБД.

Унифицировать имеющиеся модели БД

Основная идея заключается в том, чтобы из разных БД передавать информацию через какой-либо общий вид.

Взаимодействие в виде графа.

Система, в которой все схемы отображаются в виде графа и доступны для взаимодействия через графический интерфейс. В такой системе каждый узел - отдельная схема БД.

Слайд 9:

На данном слайде представлены библиотеки, которые были использованы при разработке и их назначение.

Слайд 10:

На слайде представлена диаграмма классов программы и связи между ними.

Слайд 11:

При обработке запроса можно выделить 3 части:

Первая часть - ввод запроса пользователем. Пользователь в браузере вводит запрос на получение нужных ему данных в соответствии с созданным языком запросов.

Вторая - парсер запросов. В этом элементе системы запрос пользователя разбивается на части для каждой БД.

Третья часть - классы СУБД. Каждый такой класс представляет собой СУБД для выполнения запроса, а его объекты представляют каждую БД в этой СУБД.

Слайд 12:

На слайде представлен интерфейс, от которого наследуются все классы для взаимодействия с СУБД.

Слайд 13:

Грамматика языка описывается следующим образом, показанным на слайде. Для вызова команды необходимо сначала написать названия подключения, для которого она исполняется, а затем через точку её саму. Аргументы каждой команды представлены на слайде.

Слайд 14:

Для указания столбцов необходимо через точки указать названия подключений, затем аналог таблицы для каждой СУБД и аналог столбца таблицы. Возможно, использовать сложные условия, которые имеют и или или.

Слайд 15:

На слайде представлены команды запросов к системе, которые реализуют функциональные требования.

Слайд 16:

На слайде представлен пример преобразования с созданного языка на выбранные СУБД 3 видов команд: запрос на чтение, запрос на создание данных, запрос на получение информации о индексах.

Слайд 17:

Пример на языке запросов для нескольких подключений с MongoDB представлен в верхней части картинки. Берётся вторая встречающаяся команда read, конвертируется в вид для MongoDB и затем исполняется. После результат исполнения подставляется в первую команду и после чего исполняется и она.

Слайд 18:

Разработанный метод деления на подзапросы. В строке ищется первая из возможных команд. В элемент стека, который будет потом передавать данные для исполнения, заносится подключение, для которого была вызвана команда, и сама команда. Весь аргумент заносится в элемент стека. Далее проверяется, имеется ли у команды оператор where. Если он отсутствует, то в элемент стека заносится пустой список. В случае наличия оператора where, то в условии выделяются аргументы в отдельный список и составляется список where, в который вносятся по порядку аргументы и логические и и или, если они имеются. Аргументы в нём обозначаются @. Для каждого аргумента, если они простые, заносится вместо @ каждая его часть между символами сравнения и символ сравнения между ними. Если же они составные, то для каждой части в элемент стека заносится знак % и номер создающегося элемента стека и возврат к пункту 1.

Слайд 19:

Здесь представлена блок схема, реализующий данный метод. Происходит выделение первой команды и её аргумента, затем проверка на наличие условия, если его нет, то завершение, иначе происходит выделение аргументов и затем для каждого из найденных происходит повторение, если они составные, или завершение.

Слайд 20:

На слайде представлена часть интерфейса для взаимодействия с системой.

Слайд 21:

Было проведено 3 вида тестирования - функциональное, модульное, веб-производительности. Метод для всех видов тестирования: Причина / Следствие (Cause/Effect — CE). На слайде приведён пример 3 тест-кейсов. Исходя из результатов тестирования можно сделать вывод, что все необходимые функции работают корректно.

Слайд 22:

Было проведено модульное тестирования для всех модулей. На рисунке представлен пример одного из тестов. В общем процент покрытия кода тестами составляет 80%, что означает, что основные функции программы работают корректно.

Слайд 23:

Так же было выполнено веб тестирование производительности выполнения запроса, его график и результаты представлены на слайде.

Слайд 24:

В итоге, в данной работе была разработана и протестирована с помощью 3 видов тестирования система, которая позволяет пользователю получать и анализировать информацию из разных СУБД.

Слайд 25:

Написанные статьи.

Слайд 26:

Спасибо за внимание. Готов ответить на ваши вопросы.