

Система автоматического сбора информации о работе NoSQL баз данных

Место проведения: МГТУ им. Баумана

Продолжительность: 7 минут

Научный руководитель: *Доцент, к.т.н. Виноградова Мария Валерьевна*

студент Журавлев Николай Вадимович

zhuravlevnv@student.bmstu.ru

МГТУ им. Н.Э. Баумана

Россия, Москва, 2025.02.15 – 20 мая 2025 г.



Содержание доклада

Постановка задачи

Описание системы

Программная реализация

Тестирование



Постановка задачи

↳ Концептуальная постановка задачи

Цель разработки

Создание системы автоматического сбора информации о работе NoSQL баз данных

Задачи разработки

1. Произвести обзор Nosql СУБД, использующихся в системе
2. Исследовать методы взаимодействия между различными СУБД
3. Создать модель БД, которая будет использоваться в системе
4. Разработать метод деления запроса, введённого пользователем на подзапросы и приведение к универсальному списку для каждой СУБД
5. Описать алгоритм деления запроса, введённого пользователем на подзапросы и приведение к универсальному списку для каждой СУБД
6. Разработать структуру и архитектуру системы
7. Разработать макет интерфейса программы
8. Разработать грамматику языка для запросов к системе
9. Произвести тестирование разработанной системы



Описание системы

↳ СУБД в системе

СУБД в системе и примеры поиска, добавления, удаления, обновления:

1. MongoDB

```
db.collection('name_collection').find( "elem_name" : "elem_value");  
db.collection('name_collection').insert( "elem_name" : "elem_value" );  
db.collection('name_collection').remove( "elem_name" : "elem_value" );  
db.collection('name_collection ').update( "elem_name" : " elem_value" );
```

2. Neo4j

```
MATCH (n) WHERE (n.id = 0) RETURN n;  
CREATE (node:label key1: value1, key2: value2, ..... );  
MATCH (node attribute1: 'value1') REMOVE node.attribute2 RETURN node;  
MATCH (node attribute1: 'value1') SET node.attribute2='value2' RETURN  
node.attribute1, node.attribute2;
```

3. Cassandra

```
SELECT title, MAX(price) FROM course GROUP BY title;  
INSERT INTO student (id, citizenship, first_name, last_name, age) VALUES  
(now(), 'Russia', 'Ivan', 'Ivanov', 25);  
DELETE FROM student WHERE  
id=54daf810-9aeb-11ea-b1d1-3148925e06e7;  
UPDATE student SET first_name = 'Example', last_name='Example' WHERE  
id=54daf810-9aeb-11ea-b1d1-3148925e06e7;
```



Описание системы

↳ Группы методов взаимодействия различных СУБД

Группы методов:

1. Узконаправленные системы

Системы, которые предназначены для решения проблемы в конкретной среде.

2. Ручная интеграция СУБД

Данная группа пытается вручную объединить несколько разных СУБД.

3. Создание новой СУБД

Разработать совершенно новые СУБД.

4. Унифицировать все имеющиеся модели БД

Можно каким-либо образом унифицировать все необходимые модели БД.

5. Взаимодействие в виде графа

Система, в которой все схемы отображаются в виде графа и доступны для взаимодействия через графический интерфейс.



Описание системы

↳ Модель БД

В системе используется две PostgreSQL базы данных.

Первая будет хранить данные о пользователях, пример с одной записью представлен в таблице 1.

Вторая будет хранить данные о подключениях, пример с одной записью представлен в таблице 2.

Таблица 1: Схема БД пользователей

id	login	password
0	log	pass

Таблица 2: Схема БД подключений

id	login	password	ip	port	conn_name	db_name	special	user_id
4	neo4j	87654321	localhost	7687	neo	neo4j		0



Описание системы

↳ Метод деления на подзапросы

1. В строке ищется первая из возможных команд (create, read, update).
2. В элемент стека заносится подключение, для которого была вызвана команда, и затем сама команда.
3. С начала оставшейся строки ищется конец аргумента.
4. Весь аргумент заносится в элемент стека.
5. Далее проверяется, имеется ли у команды оператор where. Если он отсутствует, то в элемент стека заносится пустой список и на место координат вставки результата записывается -1.
6. В случае наличия оператора where ищется конец его аргумента.
7. Начинается посимвольное прохождение аргумента, пока не будет встречен оператор AND или OR.
8. Символы до первого аргумента, вносятся отдельно в список where и вместо аргумента ставится @. Найденный аргумент заносится в отдельный список. Затем заносится сам оператор AND или OR.
9. Если была пройдена не вся строка, то возврат к пункту 7.
10. Для каждого аргумента в списке ищется оператор сравнения.
11. Каждый аргумент разделителя, если они простые, заносится вместо @ и сам разделитель между ними.
12. Если же они составные, то для каждой части в элемент стека заносится итерация и место, которое помечается символом % и возврат к пункту 1.



Описание системы

↳ Алгоритм деления на подзапросы. Часть 1

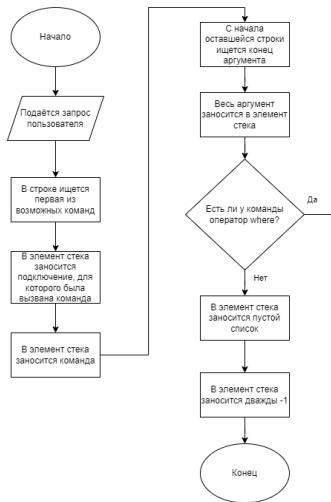


Рис. 1: Блок-схема алгоритма для деления на подзапросы. Часть 1



Описание системы

↳ Алгоритм деления на подзапросы. Часть 2

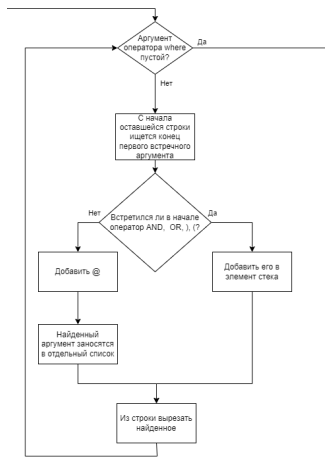


Рис. 2: Блок-схема алгоритма для деления на подзапросы. Часть 2



Описание системы

↳ Алгоритм деления на подзапросы. Часть 3

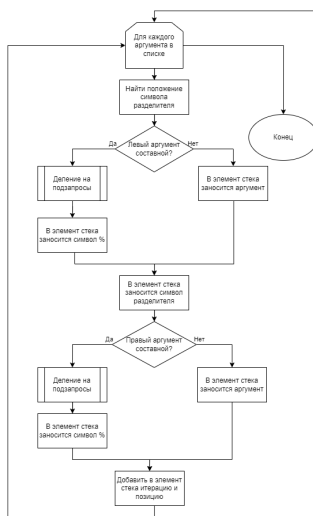


Рис. 3: Блок-схема алгоритма для деления на подзапросы. Часть 3



Программная реализация

↳ Структура и архитектура системы

Всего в системе можно выделить 3 части:

- Первая часть - ввод запроса пользователем. Пользователь в браузере вводит запрос на получение нужных ему данных в соответствии с специальным синтаксисом.
- Вторая часть - парсер запросов. В этом элементе системы запрос пользователя разбивается на части для каждой БД.
- Третья часть - классы СУБД. Каждый такой класс представляет собой возможность взаимодействия с СУБД для выполнения запроса, а его объекты представляют каждую БД в этой СУБД.

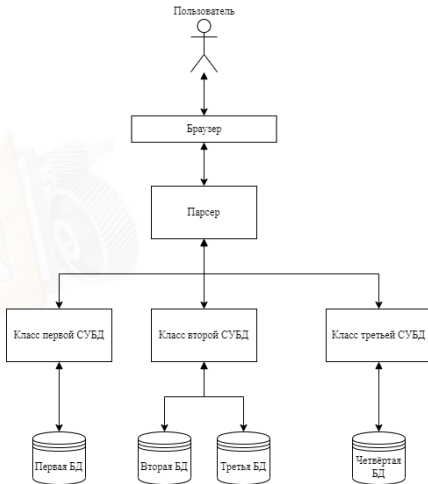


Рис. 4: Взаимодействие между элементами



Программная реализация

↳ Структура и архитектура системы

Шаблон должен иметь следующую структуру полей:

1. Логин
2. Пароль
3. IP
4. Порт
5. Название подключения
6. Особенности

Так же шаблон должен содержать следующие методы:

1. Конструктор
2. Интерпретатор
3. Деструктор

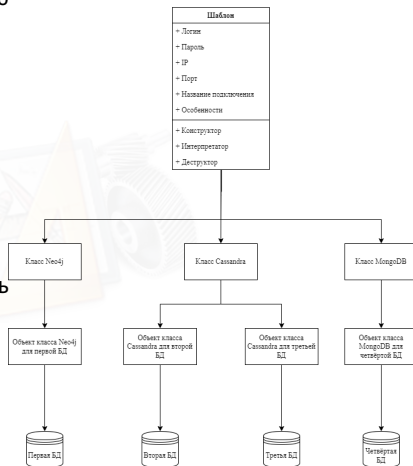
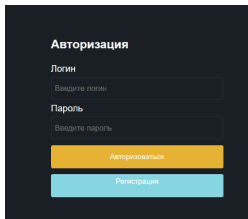


Рис. 5: Схема работы классов СУБД



Программная реализация

↳ Макет интерфейса



Авторизация

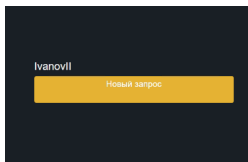
Логин

Пароль

Авторизоваться

Регистрация

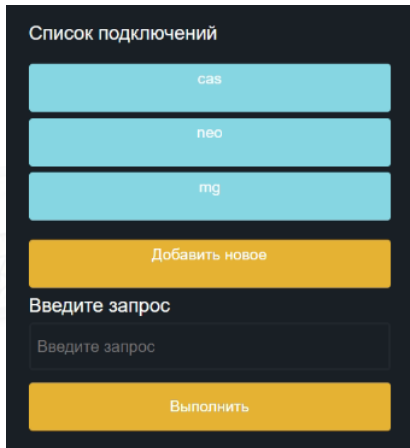
Рис. 6: Авторизация



IvanovII

Новый запрос

Рис. 7: Страница
результата



Список подключений

cas

neo

mg

Добавить новое

Введите запрос

Выполнить

Рис. 8: Основная страница



Программная реализация

↳ Грамматика запросов к системе

В системе представлены следующие операторы:

1. `x.y.z`, где `x` – название БД, `y` – название сущности, являющийся аналогом таблицы из реляционных БД, из ранее обозначенной базы данных, `z` – название сущности, являющийся аналогом столбцов из реляционных БД, из ранее обозначенной таблицы. Таким образом из БД можно получить необходимые данные.
2. `where` – оператор, в котором в скобках описывается фильтр, по которому выбираются данные из БД. В условии используются оператор доступа к данным и операторы сравнения. Сравнимыми элементами могут являться данные из БД или константы.
3. `create` – вставка данных в БД. Например,
`dbcity.create(dbcity.building.address, dbcity.building.owner, [{"Wall Street", "Stan Smith"}, {"Broadway", "John Doe"}])`.
4. `read` – чтение данных из БД. Например,
`dbcity.read(dbcity.building.address)`.
5. `update` – изменение данных в БД. Например,
`dbcity.update(dbcity.building.address, dbcity.building.owner, [{"Broadway", "John Doe"}])`
6. `delete` – удаление данных из БД. Например,
`dbcity.delete.where(dbcity.building.owner="John Doe")`.



Программная реализация

↳ Пример запроса

- Сначала идёт подключение к базе данных dbpeople.
- Парсер начинает выполнять с более меньшего подзапроса: `(dbpeople.personal_data.name).where(dbpeople.personal_data.id=1)`, где dbpeople – БД в СУБД MongoDB.
- `dbpeople.personal_data.name` преобразуется в `_id: 0, name: 1`
- `where(dbpeople.personal_data.id=1)` преобразуется в `_id: 1`
- Далее конвертируется часть запроса `dbpeople.read()` и выполняется в виде следующего запроса: `db.personal_data.find(_id: 1, _id: 0, name: 1)`.

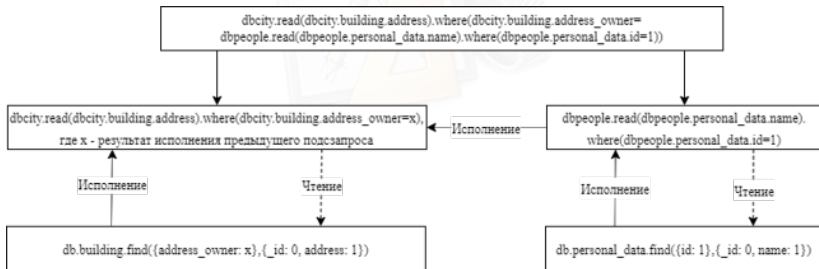


Рис. 9: Деление основного запроса на подзапросы



Тестирование

↳ Тестирование

Было проведено 2 вида тестирования - модульное и функциональное.
Пример запроса в функциональном тестировании, который выполняется проходя через все СУБД:

```
cas.read(cas.country.country_name).where(cas.country.city_name =  
neo.read(neo.city.city_name).where(neo.city.human_name =  
mg.read(mg.human.name).where(mg.human.id=0)))
```

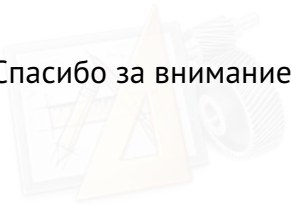
Name	Stmts	Miss	Cover
cassandradb.py	144	39	73%
mongodb.py	200	57	72%
neo4jdb.py	139	29	79%
parserdb.py	208	26	88%
template_dbms.py	7	3	57%
test_app.py	145	0	100%
TOTAL	843	154	82%

Рис. 10: Покрытие кода тестами



- В данной работе для эффективного взаимодействия между СУБД были изучены синтаксисы и грамматика для СУБД, входящих в систему.
- Была разработана и протестирована система, которая позволяет пользователю получать информацию из разных БД доступней и представлять её в более понятном виде.





Спасибо за внимание!

