

Тема: Реализация минимальной версии документно-ориентированной СУБД (MiniDBMS) с собственной хэш-таблицей и поддержкой индексов.

Цель: Получить практические навыки в реализации фундаментальных структур данных, сериализации, парсинге сложных запросов и проектировании архитектуры систем хранения данных. Изучить принципы работы индексов в базах данных.

1. Задание

Необходимо разработать приложение `no_sql_dbms`, которое реализует базовые операции документно-ориентированной СУБД. Ключевое требование - использование **собственной реализации хэш-таблицы** для хранения документов. Приложение работает в режиме одноразового выполнения команды, сохраняя данные на диск между запусками.

2. Функциональные требования

2.1. Синтаксис команд

Вставка документа:

```
./no_sql_dbms my_database insert '{"name": "Alice", "age": 25, "city": "London"}'
```

Ожидаемый ответ: Document inserted successfully.

Поиск документов:

```
./no_sql_dbms my_database find '{"age": 25}'  
./no_sql_dbms my_database find '{"name": "Alice", "city": "London"}' #  
Неявный AND  
./no_sql_dbms my_database find '{{$or}: [{"age": 25}, {"city": "Paris"}]}'  
./no_sql_dbms my_database find '{"age": {"$gt": 20}}'  
./no_sql_dbms my_database find '{"name": {"$like": "Ali%}}'  
./no_sql_dbms my_database find '{"city": {"$in": ["London", "Paris"]}}'
```

Удаление документов:

```
./no_sql_dbms my_database delete '{"age": 25}'  
./no_sql_dbms my_database delete '{"name": {"$like": "A%"}'
```

Создание индекса (задание со звездочкой):

```
./no_sql_dbms my_database create_index age
```

2.2. Обязательные операторы

Логические операторы:

- `$and` - неявный (через запятую в JSON) и явный
- `$or` - явный оператор для объединения условий

Операторы сравнения:

- `$eq` - равенство (используется по умолчанию)
- `$gt` - больше (greater than)
- `$lt` - меньше (less than)
- `$like` - строковый поиск с поддержкой wildcards (% - любая строка, _ - один символ)
- `$in` - проверка принадлежности к массиву значений

3. Техническая реализация

3.1. Обязательное требование: Собственная HashMap

Студенты должны реализовать свою хэш-таблицу для хранения документов в коллекциях.

Требования к реализации:

- **Хэш-функция:** Реализовать собственную хэш-функцию для строковых ключей
- **Разрешение коллизий:** Использовать метод цепочек (chaining)
- **Базовые операции:** `put(key, value)`, `get(key)`, `remove(key)`, `items()`
- **Динамическое расширение:** Увеличение размера таблицы при достижении коэффициента загрузки

3.2. Архитектура хранения данных

- **База данных:** Представляет собой директорию с файлами
- **Коллекция:** Группа документов, хранящаяся в отдельном файле

- **Документ:** JSON-объект с автоматически генерируемым полем `_id`
- **Хранение:** Каждая коллекция представлена HashMap, где ключ - `_id` документа, значение - сам документ

3.3. Алгоритмы обработки запросов

Поиск документов:

1. Загрузка коллекции из файла в HashMap
2. Последовательный перебор всех документов
3. Проверка каждого документа на соответствие условиям запроса
4. Применение логических операторов (`$and` , `$or`)
5. Выполнение операторов сравнения для каждого поля

Удаление документов:

1. Поиск документов, соответствующих условию (аналогично `find`)
2. Удаление найденных документов из HashMap
3. Сохранение измененной коллекции на диск

3.4. Персистентность данных

- **Сериализация:** Сохранение HashMap в JSON-файл при завершении операций
 - **Загрузка:** Восстановление HashMap из JSON-файла при старте
 - **Формат хранения:** Читаемый JSON для отладки
-

4. Задание со звездочкой: B-Tree индекс

Реализация механизма индексации для ускорения поиска по определенным полям.

4.1. Требования к B-Tree индексу

- **Структура:** Реализовать B-Tree с настраиваемым порядком
- **Операции:** Вставка, поиск, диапазонный поиск
- **Хранение:** Отдельный файл для каждого индекса

4.2. Механизм работы индекса

Создание индекса:

- Сканирование всех документов коллекции
- Построение B-Tree, где ключ - значение индексируемого поля, значение - массив `_id` документов
- Сохранение индекса в отдельный файл

Поиск с использованием индекса:

1. Загрузка B-Tree индекса и коллекции в память
2. Поиск в индексе по условию запроса (точечный поиск для `$eq`, диапазонный для `$gt / $lt`)
3. Получение массива `_id` соответствующих документов
4. Извлечение полных документов из HashMap коллекции по найденным `_id`

Поддержка операторов в индексе:

- `$eq` - точечный поиск в B-Tree
 - `$gt`, `$lt` - диапазонный поиск
 - `$in` - множественный точечный поиск
-

5. Рекомендации по выполнению

1. **Этап 1:** Реализовать и протестировать HashMap
2. **Этап 2:** Реализовать базовые операции вставки и простого поиска
3. **Этап 3:** Добавить операторы сравнения и логические операторы
4. **Этап 4:** Реализовать операцию удаления
5. **Этап 5 (оpционально):** Реализовать B-Tree индекс и интеграцию с поиском

Тестирование: Для проверки рекомендуется создать тестовые сценарии с различными типами данных и сложными запросами.

Материалы:

- <https://yandex.cloud/ru/blog/posts/2022/10/nosql>
https://en.wikipedia.org/wiki/Hash_table
<https://neerc.ifmo.ru/wiki/index.php?title=B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>
<https://www.geeksforgeeks.org/mongodb/what-is-a-mongodb-query/>