|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА, ИССКУСТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_Методы оптимизации структуры баз данных\_\_ \_\_\_\_\_\_\_\_\_NoSQL типа ключ-документ\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_ИУ5-33М\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_В.В.Пенегина\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Ю.Е.Гапанюк\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2023 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_Методы оптимизации структуры баз данных NoSQL типа ключ-документ\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_ИУ5-33М\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пенегина Вероника Владимировна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_исследовательская\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

***Техническое задание \_****Сбор информации о базах данных NoSQL, способов оптимизации их структуры, индексирования, шардинга, денормализации данных*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_14\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_1\_ » \_\_сентября\_\_ 2023 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**Ю.Е. Гапанюк\_\_**\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_В.В. Пенегина\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[Введение 4](#_Toc138443938)

[Постановка задачи 4](#_Toc138443939)

[Понятия NoSql и ключевые особенности 5](#_Toc138443940)

[Шардинг и денормализация данных 5](#_Toc138443941)

[Методы оптимизации структуры нераспределенных баз данных NoSQL 7](#_Toc138443942)

[Список используемой литературы 14](#_Toc138443943)

# **Введение**

В современном мире, где данные играют ключевую роль в бизнес-процессах, эффективное управление базами данных является одним из основных факторов успеха. С появлением NoSQL баз данных, структуры данных стали более гибкими и масштабируемыми, что позволяет хранить и обрабатывать большие объемы информации. Однако, чтобы достичь максимальной производительности и эффективности, необходимо правильно оптимизировать структуру баз данных типа ключ-документ. В данной статье мы рассмотрим методы оптимизации структуры баз данных NoSQL типа ключ-документ, которые помогут улучшить производительность и снизить нагрузку на сервер.

# **Постановка задачи**

Целью научно-исследовательской работы является изучение совокупности формализованных методов консолидации данных из реляционной базы данных с оптимизацией ее схемы и запросов для ускорения обработки и хранения данных.

В соответствии с поставленной целью в научно-исследовательской работе решались следующие задачи:

1. Изучение баз данных NoSQL;
2. Анализ методов оптимизации и их структуры;
3. Изучение способов шардинга и денормализации данных.

Для решения поставленных задач использовались теория баз данных, теория множеств, теория графов, теории функциональных зависимостей, реляционная алгебра. Для разработки программного обеспечения и тестирования разработанных методов использованы языки программирования Python, среда разработки Microsoft Visual Studio Community, Visual studio code, СУБД MongoDB.

# **Понятия NoSql и ключевые особенности**

NoSQL (Not Only SQL) - это семейство баз данных, отличающихся от реляционных БД и предоставляющих различные модели данных и возможности. Основные понятия включают гибкую схему данных, горизонтальное масштабирование, горизонтальное и вертикальное фрагментирование, отсутствие жестких требований к согласованности и распределенную архитектуру. При работе с NoSQL-базами данных возникают проблемы согласованности данных. Одна из проблем - окно несогласованности, когда данные могут быть доступны с некоторым временным разрывом. Для решения этих проблем используются оптимистический и пессимистический подходы. Теорема CAP говорит о невозможности одновременного обеспечения согласованности (C), доступности (A) и устойчивости к разделению (P) в распределенных системах. Кворумы и компромиссы используются для достижения баланса между согласованностью и доступностью.

БД типа "ключ-значение" в NoSQL представляет собой хранилище, в котором каждое значение связано с уникальным ключом. Модель данных проста и плоская, а функции базы данных ограничены операциями чтения и записи по ключу. Реализация транзакций может быть ограничена или отсутствовать. Масштабирование происходит путем разделения данных по ключу, а согласованность может быть слабой.

# **Шардинг и денормализация данных**

Шардинг и денормализация данных - это два популярных подхода к управлению большими объемами данных. Оба метода используются для оптимизации работы с данными и ускорения процессов обработки информации. В этой главе мы рассмотрим, что такое шардинг и денормализация данных, как они работают и какие преимущества они могут предоставить вашей компании.

Шардинг - это процесс разбиения большой базы данных на несколько более мелких фрагментов, называемых шардами. Каждый шард содержит только часть данных, которые нужны для выполнения конкретной операции. Например, если у вас есть база данных с информацией о клиентах, вы можете разбить ее на несколько шардов, каждый из которых будет содержать данные только о клиентах из конкретного региона или страны.

Шардинг может быть реализован на уровне приложения или базы данных. На уровне приложения это означает, что приложение само заботится о том, как данные хранятся и обрабатываются. На уровне базы данных это означает, что база данных разбивается на несколько физических серверов, каждый из которых содержит только часть данных.

Шардинг может предоставить несколько преимуществ вашей компании. Во-первых, он может улучшить производительность приложения, так как каждый шард может обрабатывать только часть данных, что уменьшает нагрузку на сервер. Во-вторых, шардинг может обеспечить более высокую доступность данных, так как если один из серверов выходит из строя, другие серверы могут продолжать работу без проблем.

Денормализация - это процесс изменения структуры базы данных таким образом, чтобы данные хранились в виде дубликатов в нескольких таблицах. Например, если у вас есть таблицы с информацией о клиентах и заказах, вы можете создать дополнительную таблицу, которая будет содержать информацию о клиентах и их заказах. Это позволит быстрее получать данные о клиентах и их заказах, так как они будут храниться в одной таблице.

Денормализация может быть реализована на уровне приложения или базы данных. На уровне приложения это означает, что приложение само заботится о том, как данные хранятся и обрабатываются. На уровне базы данных это означает, что данные хранятся в виде дубликатов в нескольких таблицах.

Денормализация может ускорить процесс получения данных, так как данные будут храниться в одной таблице. Также, денормализация может уменьшить нагрузку на сервер, так как приложение не будет запрашивать данные из разных таблиц. В-третьих, денормализация может обеспечить более высокую доступность данных, так как если один из серверов выходит из строя, другие серверы могут продолжать работу без проблем.

# **Методы оптимизации структуры нераспределенных баз данных NoSQL**

В условиях повышенного роста данных и требований к современным информационным системам является актуальной проблема трансляции данных между различными источниками данных. Однако различия между методами хранения данных в этих двух форматах SQL и NoSQL делают процессы трансляции, трансформации и консолидации данных затруднительными. Одной из проблем при трансляции данных из формата SQL БД в формат NoSQL БД типа «ключ-документ» является сопоставление коллекций NoSQL таблицам реляционных баз данных. Для других типов NoSQL БД проблемой является отсутствие структуры БД и особенности языка запросов. Если для реляционных баз данных существуют методы формализованного построения отношений на основе заданных свойств объектов и функциональных зависимостей между ними, то для NoSQL и NewSQL такого формализованного аппарата не существует. И, например, при трансляции реляционной базы данных в формат документной БД MongoDB, всегда есть проблема как этот перевод осуществлять наиболее эффективно для последующей работы с данными.

При этом возможны следующие варианты изменения формата:

1) каждой таблице в реляционной базе данных поставить в соответствие отдельную коллекцию документов в MongoDB;

2) из всех таблиц реляционной базы данных сделать одну коллекцию документов в MongoDB;

3) создать такой набор коллекций документов в MongoDB, чтобы они наиболее полно подходили под выполняемые запросы. Данный вариант приводит к проблеме выбора числа коллекций и состава каждой коллекции.

Входными данными является совокупность свойств объектов, хранимых в базе данных. Если задача заключается в трансляции реляционной базы данных в формат MongoDB, то в качестве совокупности свойств объекта выступает множество полей всех таблиц. По определению множества среди его элементов не может быть дубликатов.

Пусть 𝑇𝑟 – это таблица реляционной базы данных, где 𝑟 – номер таблицы, 𝑟 = 1...𝑘, 𝑘 –число таблиц. Пусть 𝑇(𝑟,𝑗) – это поле в таблице 𝑇𝑟, где 𝑗 – номер поля, 𝑗 = 1..𝑟𝑛, 𝑟𝑛 – число полей в 𝑟-й таблице. Тогда множество полей одной таблицы – это множество вида:

𝑇𝑟 = {𝑇𝑟,𝑗 , 𝑗 = 1,2,..,𝑟𝑛} (2.1)

Множество всех полей реляционной базы данных будет определяться по формуле (2.1):

𝑀 = {𝑇𝑟,𝑗 ,𝑟 = 1,2,...,𝑘|𝑇𝑟,𝑗 ̸= 𝑇𝑞,𝑖,∀𝑟,𝑞 ⩽ 𝑘,𝑗 ⩽ 𝑟𝑛,𝑖 ⩽ 𝑞𝑛} (2.2)

где 𝑟 – номер таблицы, 𝑟 = 1...𝑘, 𝑘 –число таблиц, 𝑗 – номер поля в таблице, 𝑗 = 1...𝑟𝑛, 𝑟𝑛 – число полей в 𝑟-й таблице.

Длина множества – это количество его элементов. Обозначение: |𝑀| – длина множества. Совокупность множеств полей, входящих в запрос:

𝑆𝑖 = {𝑇𝑟,𝑗 , 𝑟 ⩽ 𝑘, 𝑗 ⩽ 𝑟𝑛} (2.3)

где 𝑖 – номер запроса (𝑖 = 1,2, . . . 𝑚), 𝑚 – число запросов к базе данных. Выходные данные: Выходными данными является совокупность множеств коллекций документов с заданными полями: 𝑉𝑖 = {𝑇𝑟,𝑗 , 𝑟 ⩽ 𝑘, 𝑗 ⩽ 𝑟𝑛} удовлетворяющие условиям: 𝑉1 ∩ 𝑉2 ∩ 𝑉3 = ∅ (2.4) 𝑉1 ∪ 𝑉1 ∪ 𝑉3 = 𝑀 = ⋃︁𝑘 𝑟=1 𝑇𝑟 (2.5) (∀𝑆𝑖)(∃𝑉𝑗 )(𝑆𝑖 ∈ 𝑉𝑗 ,𝑆𝑖 ∈/ 𝑉𝑖 ,𝑖 ̸= 𝑗) (2.6) где 𝑖 – номер коллекции (𝑖 = 1,2,...𝑙), 𝑙 – число коллекций. Метод формирования коллекций для базы данных в формате MongoDB:

Шаг 1. Создать множества полей таблиц и запросов по формулам (2.2 - 2.3).Число множеств будет равно числу таблиц.

Шаг 2. Выбрать поля, которые не участвуют в запросах. Для этого: 2.1. Для каждого поля составить множества запросов, в которых это поле участвует в любой части конструкции: 𝑇 ′ 𝑟,𝑗 = {𝑆𝑖 , 𝑖 = 1....𝑝, 𝑝 ⩽ 𝑚|𝑇𝑟,𝑗 ∈ 𝑆𝑖} где 𝑚 – число запросов, 𝑟 – номер таблицы, 𝑟 = 1...𝑘, 𝑘 –число таблиц, 𝑗 – номер поля в таблице, 𝑗 = 1...𝑟𝑛, 𝑟𝑛 – число полей в 𝑟-й таблице. 2.2. Все поля 𝑇𝑟,𝑗 , для которых длина множества |𝑇 ′ 𝑟,𝑗 | = 0, могут быть включены в единую коллекцию MongoDB: 𝑉1 = {𝑇𝑟,𝑗 , 𝑟 ⩽ 𝑘, 𝑗 ⩽ 𝑟𝑛||𝑇 ′ 𝑟,𝑗 | = 0} 42 где 𝑚 – число запросов, 𝑟 – номер таблицы, 𝑟 = 1...𝑘, 𝑘 –число таблиц, 𝑗 – номер поля в таблице, 𝑗 = 1...𝑟𝑛, 𝑟𝑛 – число полей в 𝑟-й таблице.

Шаг 3. Выбрать поля, которые участвуют только в одном запросе, т.е. |𝑇 ′ 𝑟,𝑗 | = 1. Если для любого 𝑇𝑟,𝑗 ∈ 𝑆𝑖 выполняется условие |𝑇 ′ 𝑟,𝑗 | = 1, то все поля 𝑇𝑟,𝑗 множества 𝑆𝑖 должны войти в новую коллекцию:

𝑉𝑝 = {𝑇𝑟,𝑗 , 𝑟 ⩽ 𝑘, 𝑗 ⩽ 𝑟𝑛|𝑇𝑟,𝑗 ∈ 𝑆𝑖& |𝑇𝑟,𝑗 | = 1},𝑝 > 1 (2.7)

Шаг 4. Убрать из рассмотрения все поля, которые вошли в коллекции 𝑉1 и 𝑉𝑝 на шагах 1-3.

Шаг 5.1. Составить коллекции из полей, к которым обращается несколько запросов. Составить множество рассматриваемых запросов: 𝐼 = {𝑆𝑖}, 𝑖 = 1...𝑚, где 𝑚 – число рассматриваемых запросов. Составить попарные пересечения множеств 𝑆𝑖 ∈ 𝐼. 𝑆 ′ 𝑘 = 𝑆𝑖 ∩ 𝑆𝑗 ,∀𝑖 ̸= 𝑗;𝑖,𝑗 = 1,2,...,|𝐼|; 𝑘 = 1,2,...,𝐶2 |𝐼| где 𝐶 2 |𝐼| = |𝐼|! |(|𝐼|−2)|!2! = |𝐼|(|𝐼|−1) 2 . Для полученных не пустых пересечений составить множество 𝑃 из запросов, входящих в эти пересечения: 𝑃 = {𝑆 ′ 𝑖 ||𝑆 ′ 𝑖 | ̸= 0,∀𝑖 ⩽ 𝑘,𝑘 = 1,2,...,𝐶2 |𝐼| }. Найти разность множеств: 𝐼 − 𝑃. Если 𝐼 −𝑃 ̸= ∅, то новая коллекция будет состоять из всех полей, входящих в запросы разности множеств 𝐼 − 𝑃.

Шаг 5.2. Положить 𝐼 = 𝑃 5.6. Из полученных пересечений найти новые не пустые пересечения по правилу: 𝑖𝑓(∃(𝑆𝑖 ∩ 𝑆𝑗 )&∃(𝑆𝑗 ∩ 𝑆𝑘)),𝑓 𝑖𝑛𝑑(𝑆𝑖 ∩ 𝑆𝑗 ∩ 𝑆𝑘) или в общем виде: 𝑆 ′′ 𝑘 = 𝑆 ′ 𝑖 ∩ 𝑆 ′ 𝑗 ,𝑖𝑓∃𝑆𝑚|(𝑆𝑚 ∈ 𝑆 ′ 𝑖&𝑆𝑚 ∈ 𝑆 ′ 𝑗 ,𝑖 = 𝑗); 𝑖,𝑗 = 1,2,...,|𝐼|; 𝑘 = 1,2,...,𝐶2 |𝐼|

Шаг 5.3. Повторить шаги 5.1-5.2 до тех пор, пока не останется одно пересечение, т.е. длина множества 𝐼 не станет равной 1: |𝐼| = 1

Шаг 6. В новое отношение включить все поля, вошедшие в последнее единственное пересечение и не вошедшие в другие пересечения: 𝑉𝑝+1 = 𝑀 − ⋃︀ 𝑝 𝑖=1 𝑉𝑖

Шаг 7. Конец метода. Пример применения метода: Пусть дана база данных, состоящая из 5 отношений:

1. Т1(𝑇11, 𝑇12, 𝑇13, 𝑇14, 𝑇15)

2. Т2(𝑇21, 𝑇22, 𝑇23, 𝑇24, 𝑇25, 𝑇26)

3. Т3(𝑇31, 𝑇32, 𝑇33, 𝑇34, 𝑇35, 𝑇36)

4. Т4(𝑇41, 𝑇42, 𝑇43, 𝑇44, 𝑇45, 𝑇46)

5. Т5(𝑇51, 𝑇52, 𝑇53, 𝑇54)

Пусть даны запросы к базе данных: 1. Select sum(𝑇14-𝑇12), 𝑇11 From T1 Where 𝑇15>a and 𝑇15a and 𝑇26a and 𝑇15a and 𝑇26a and 𝑇15a and 𝑇26=a and 𝑇34<=b; Order by 𝑇34 7. Select distinct 𝑇31,𝑇32 From T3, T4 Where 𝑇31=a and 𝑇33=𝑇41;

Разработанный метод определения коллекций может быть применен:

1) для создания эффективной структуры новой базы данных типа «ключ-документ»;

2) для трансляции данных из реляционной базы данных в базу данных типа «ключ-документ»;

3) для трансляции данных из произвольной базы данных в базу данных типа «ключ-документ»;

4) для консолидации баз данных.

Есть и другие возможности для применения данного метода, такие как создание временных коллекций и т.п. Принцип применения предлагаемого метода для разного рода преобразования структуры базы данных приведен на рисунке 1. Для определения необходимости создания вложенных документов лучше, если схема будет приведена изначально к нормальной форме с помощью реляционной алгебры. Это позволит найти компромисс между быстродействием запросов и минимальным объемом памяти для хранения данных. Если БД изначально не была реляционной, то привести ее к реляционной форме чаще всего можно. Правда при этом могут получиться сильно разреженные отношения. Но, в данном случае – разреженность не играет никакой отрицательной роли, т.к. нормальная форма – это всего лишь промежуточная форма. Она необходима для установления связей между отношениями, которые впоследствии будут учтены при создании вложенные документов.

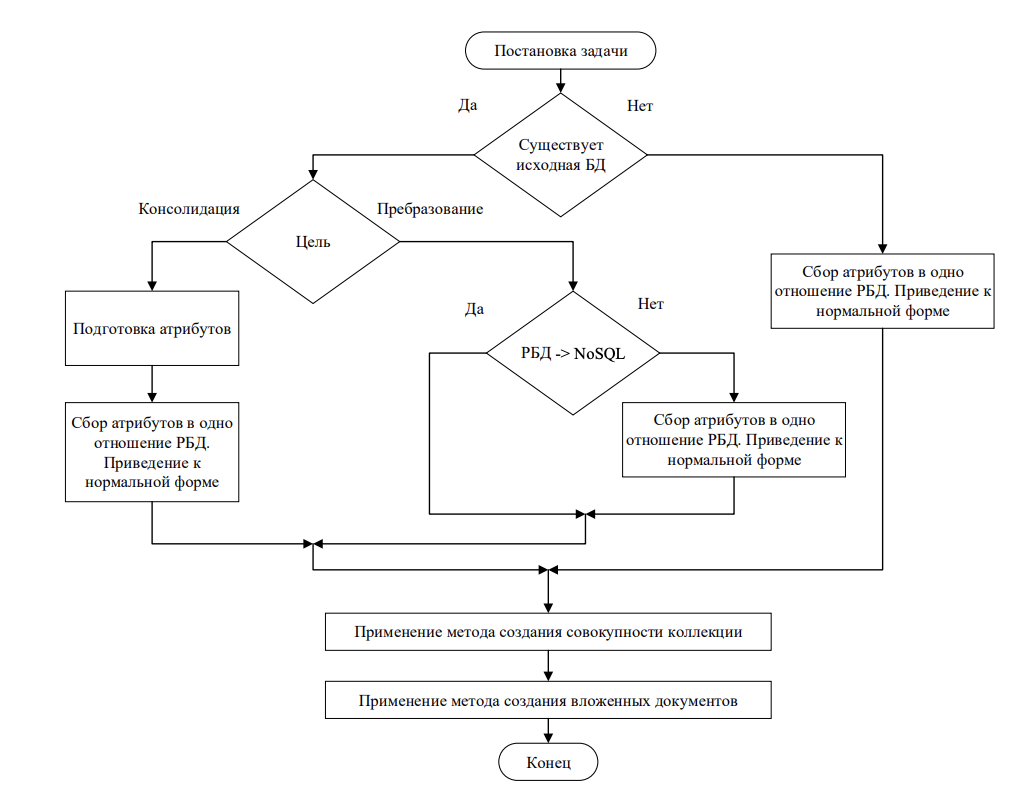


Рисунок 1 — Применение метода определения коллекций в БД типа ключ-документ

Таким образом, если исходная БД изначально не была реляционной, то преобразование ее схемы должно пройти по принципу: NoSQL-РБД-NoSQL. И последнее надо отметить, что данные в новую структуру переносятся в последнюю очередь и делается это как правило с помощью программных надстроек или с помощью алгебры кортежей, что частично описано в [1].

В основе метода оптимизации схемы документных баз данных с учетом структуры выполняемых запросов для случая распределенной базы данных лежат методы для централизованных баз данных, а также методы оптимизации графа информационных зависимостей. Запрос – это набор инструкций, которые описывают порядок действий исполнителя для достижения определенного результата. После появления возможности параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Можно сказать, что запрос к базе данных – это определение порядка и условий выбора данных, а процесс построения запроса – это разложение запроса на элементарные подзапросы. Под элементарным подзапросом будем понимать конструкцию Select без вложенных подзапросов. Если запрос содержит внутри себя элементарный запрос (подзапрос), то будем называть его родительским, а подзапрос – дочерним. Под информационным графом (information graph) будем далее подразумевать ориентированный ациклический мультиграф, вершины которого соответствуют элементарным запросам внутри основного запроса, а дуги - передаче данных между ними.

Принцип эффективной организации параллельной обработки запроса подразумевает поиск независимых друг от друга по данным подзапросов. При построении новой структуры запроса важно, чтобы сохранялись такие свойства запроса, как точность вычислений и устойчивость запроса. А для этого важна информация о том, у каких подзапросов входные данные являются результатами выполнения других подзапросов. Эти связи отражает информационный граф. Важную роль при этом играет понятие параллельной формы графа. При модификации параллельной формы информационного графа не изменяется структура графа, но только его проекция на плоскости. За счет этого новый запрос, построенный по модифицированной параллельной форме информационного графа, является эквивалентным исходному запросу. Но, стоит отметить, что при таком подходе может не достигаться глобальный экстремум в оптимизации решения, поставленной задачи по определенным параметрам, например, по времени. При этом подходе будет найден самый оптимальный запрос среди всех эквивалентных запросов с учетом связей по данным между подзапросами. Если убрать это ограничение «связь по данным», то вполне возможно, что удастся построить еще более эффективный запрос, но тогда он уже не будет эквивалентным и поэтому придется отдельно проверять его численные характеристики и тестировать. Для построения параллельного плана запросов в первом приближении можно использовать метод распараллеливания классических алгоритмов, основанный на списках смежности информационного графа алгоритма [2]. В отличии от классических алгоритмов информационный граф запроса строится намного проще и имеет намного меньше вершин.

**Заключение**

Проанализированы методы оптимизации структуры нераспределенных баз данных NoSQL. Установлено, что если исходная БД изначально не была реляционной, то преобразование ее схемы должно пройти по принципу: NoSQL-РБД-NoSQL. Данные в новую структуру переносятся в последнюю очередь и делается это как правило с помощью программных надстроек или с помощью алгебры кортежей.

Исследования, проведенные в рамках данной работы, показали, что все методы оптимизации информационного графа по различным параметрам (времени, вычислительным узлам, объему межпроцессных передач и т.д.), разработанные для классических алгоритмов, полностью подходят для запросов.

# **Список используемой литературы**

1. Система управления базами данных [Электронный ресурс] // Википедия : Свободная энциклопедия. URL : <https://ru.wikipedia.org/wiki/Система_управления_базами_данных> (дата обращения 30.02.2023)

2. Нормализация и денормализация базы данных, нормальные формы [Электронный ресурс] // Patches : IT Community. URL : <https://oracle-patches.com/db/нормализация-и-денормализация-базы-данных-нормальные-формы> (дата обращения 31.03.2023)

3. Кузнецов С. Д., Посконин А. В. Системы управления данными категории NoSQL // Программирование. — 2014. — No 6. — С. 34–47.

4. Optimization of relational databases schemas by means of n-tuple algebra / J. Shichkina, A. Degtyarev, B. Kulik, A. Fridman // AIP Conference Proceedings. — 2017. — Vol. 1863, no. 1.

5. Dash D., Polyzotis N., Ailamaki A. CoPhy: A Scalable, Portable, and Inter¬ active Index Advisor for Large Workloads // Proc. VLDB Endow. — 2011. — Vol. 4, no. 6. — Pp. 362–372.

6. Gabriel P. L., Felipe M. Automated Database Indexing using Model-free Reinforcement Learning. — URL: https://arxiv.org/abs/2007.14244 (дата обращения 28.04.2023).

7. Меликов А. В. Применение теории множеств для организации данных исходной реляционной базы данных // Прикаспийский журнал: управление и высокие технологии. — 2011. — No 4. — С. 16–22.