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 и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод миграции данных из реляционной в документо-ориентированную базу данных с использованием частотного и семантического анализа»

Студент ИУ7-84Б (Группа) _____ (Подпись, дата) Шингаров И. Д. (И. О. Фамилия)

Руководитель ВКР _____ (Подпись, дата) Вишневская Т. И. (И. О. Фамилия)

Нормоконтролер _____ (Подпись, дата) Мальцева Д. Ю. (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 59 с., 19 рис., 4 табл., 23 источн., 1 прил.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

—

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

—

ВВЕДЕНИЕ

1 Аналитический раздел

В данном разделе производится анализ предметной области, дается краткая характеристика различных типов баз данных и излагаются области применения миграции баз данных. Приводится описание и сравнение методов миграции данных из реляционной в документо ориентированную базу данных. Производится постановка задачи в виде IDEF0-диаграммы.

1.1 Базы данных

База данных (БД) – совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживат одну или более областей применения.

Система управления базами данных (СУБД) – это комплекс программных средств, котрый предоставляет возможность обращаться к базе данных для некоторой группы пользователей.

Задачи СУБД:

— управление данными во внешней памяти;

— управление транзакциями;

— журналирование;

— поддержка языка запросов;

СУБД можно классифицировать по используемой для организации хранения модели данных. Среди разных моделей данных существуют такие как: иерархическая, сетевая, объектно-ориентированная, реляционная, ассоциативная и другие.

Реляционная модель данных, является наиболее широко используемой. На таблице 1.1 представлен рейтинг СУБД по популярности на момент марта 2023 года с указанием используемой модели данных [1].

Таблица 1.1 – Рейтинг популярности СУБД

Позиция	СУБД	Модель данных	Рейтинг
1	Oracle	Реляционная	1232.64
2	MySQL	Реляционная	1172.46
3	MsSQL	Реляционная	920.09
4	PostgreSQL	Реляционная	617.90
5	MongoDB	Документо-ориентированная	436.61
6	Redis	Ключ-значение	168.13

1.2 Реляционные базы данных

Идея реляционной модели была предложена Э. Ф. Коддом в 1970 году [2]. В реляционной модели данные хранятся в виде таблиц связанных отношениями. Отношения могут описаны логически в форме «один-к-одному» «один-к-многим» «многие-к-многим». Визуально представить такую модель можно с помощью нотации Чена, предложенной для описания моделей «сущность-связь» в 1976 году [3].

Данные в реляционной модели должны удовлетворять следующим свойствам:

1. каждая строка таблицы отображает кортеж или запись;
2. все значения атрибутов атомарны;
3. все строки уникальны;
4. порядок строк не значим;
5. порядок колонок не значим.

1.2.1 Основные понятия

Рассмотрим основные понятия реляционной модели данных.

Ключ

Понятие ключа является одним из основных для РСУБД. Ключ позволяет идентифицировать запись и облегчает создание отношения между сущностями. Также засчет ключей достигается важное свойство реляционной модели – уникальность записей.

Есть три основных типа ключей:

— Потенциальный ключ – используется для того, чтобы определить сущность без обращения к другим данным таблицы. Должен представлять собой минимальный набор атрибутов для идентификации сущности. Потенциальных ключей может быть несколько, они могут быть как простыми так и составными.

— Первичный ключ – ключ который используется для идентификации записи в таблице. Первичный ключ может быть только один и выбирается из множества потенциальных ключей. Засчет первичного ключа обеспечивается уникальность и отсутствие избыточности данных. На рисунке 1.1 поля «ID» у сущностей «человек» и «собака» являются первичными ключами.

— Внешний ключ – атрибут или набор атрибутов, который ссылается на соответствующий первичный ключ другой сущности для того, чтобы однозначно определить эту другую сущность. На рисунке 1.1 поле «хозяин» является внешним ключом сущности «собака» и ссылается на сущность «человек» определяя связь «один-к-многим» между человеком и собакой.

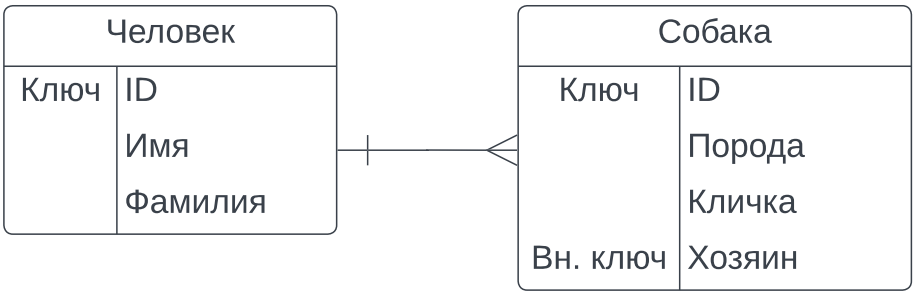

Рисунок 1.1 – Отношение с использованием первичного и внешнего ключа

Нормализация данных

Нормализация – это набор правил, согласно которым необходимо проектировать сущности или таблицы в реляционной модели для определения отношений через первичные и внешние ключи. Эти правила нужны для решения проблемы сложных доменов описанной Э. Ф. Коддом [2]. Нормализация представляет собой процесс разбиения таблиц на подтаблицы облегчая дальнейшее взаимодействие с данными в базе.

Можно выделить следующие приемущества нормализации:

— устранение аномалий при обновлении.

— оптимизация запросов.

— обеспечение целостности данных.

— гибкость и производительность при аггрегации, сортировках

В 1979 Кодд ввел три типа нормализации, которые были названы нормальными формами [4]:

— Первая нормальная форма (1NF) – согласно первой нормальной формы не должно присутствовать повторяющихся групп. Если различные колонки таблицы содержат одинаковый тип информации, тогда таблица не соответствует 1NF. Данные должны быть разбиты на минимально допустимые фрагменты. Согласно Кодду все данные в таблице должны быть неделимы.

— Вторая нормальная форма (2NF) – вторая нормальная форма наследует требования 1NF. В 2NF все неключевые атрибуты должны быть функционально зависимыми от первичного ключа.

— Третья нормальная форма (3NF) – третья нормальная форма удовлетворяет всем требованиям 1NF и 2NF. В 3NF необходимо, чтобы все неключевые атрибуты должны зависеть от первичного ключа и не зависеть от неключевых атрибутов.

1.2.2 Язык запросов

Реляционные базы данных используют сильно-структурированный язык для доступа к базе и последующего получения данных. Этот язык получил название «Structured Query Language» (SQL). Данный язык был разработан Дональдом Чемберлейном и Реймондом Бойсом в 70е для получения доступа к реляционным базам данных [5].

SQL быстро заработал популярность и в дальнейшем большинство разработчиков РСУБД внедрили его в свои продукты. Основной причиной популярности языка стало то, что он значительно облегчал разработчикам взаимодействие с базами данных, что в свою очередь сокращало время на разработку и стоимость программного продукта в целом. В следствие популярности Американский Национальный Институт Стандартизации (ANSI) разработал стандарт SQL – ANSI SQL. Несмотря на стандарт языка, множество СУБД используют собственные модификации языка SQL.

1.2.3 ACID Свойства

Важный набор понятий, гарантирующий безопасность и надежность транзакции известен как ACID свойства [6]. Логическая единица работы которая производится внутри последовательности операций называется транзакцией. Для того, чтобы обеспечить безопасную, надёжную и согласованную транзакцию, логическая единица работы должны удовлетворять четырем свойствам – неразрывность, правильность, изоляция и устойчивость. Аббревиатура ACID соответственно сформирована из первых букв слов Atomicity, Consistency, Isolation, Durability.

— Неразрывность (англ. Atomicity) – “Транзакции неразрывны (Всё или ничего)”. В транзакции либо выполняются все необходимые операции, либо изменения не производятся.

— Правильность (англ. Consistency) – “Транзакции переводят базу из одного правильного состояия в другое”. При этом правильность не обязана обеспечиваться на всех этапах транзакции.

— Изолированность (англ. Isolation) – “Транзакции изолированы друг от друга”. Промежуточное состояние транзакции не может быть получено любой другой транзакцией. Изоляция необходима для того чтобы поддерживать согласованность между транзакциями. Изменения производимые параллельным выполнением транзакций также должны быть изолированы друг от друга.

— Устойчивость (англ. Durability) – После завершения успешной транзакции изменения должны быть зафиксированы в системе даже в случае выхода системы из строя.

1.3 Нереляционные базы данных

Нереляционные базы данных принято называть NoSQL от «Not Only SQL», так как они разрабатывались для распределенного хранения данных и тем самым они выходят за рамки ограничений накладываемых реляционной моделью данных, для которой был разработан язык SQL. NoSQL базы данных проектируются с учетом потребностей в распределенных и высоконагруженных облачных вычислениях на больших объемах данных. Также зачастую к ним предъявляется треобвание иметь гибкую схему для эффективного хранения неструктурированных данных [7]. Можно выделить такие ключевые особенности NoSQL баз данных:

1. Возможность горизонтального масштабирования.
2. Возможность партиционирования и распределения по множеству серверов.
3. Сравнительно ненадёжный параллелизм в сравнении с ACID.
4. Схожий с SQL синтаксис. Простой язык запросов.
5. Возможность динамически менять количество атрибутов.
6. Эффективное использование ОЗУ и распределенных индексов для хранения данных.

Горизонтальное масштабирование, репликация и распределение данных по нескольким серверам дают возможность более быстро осуществлять операции чтения и записи данных. Однако, NoSQL не удовлетворяет свойствам ACID, которые обеспечивают правильность параллельных транзакций. Веб-приложения в основном выполняются в распределенной среде, где основным требованием является возможность к масштабированию и согласно «CAP-теореме» для распределенной системы не предоставляется возможным обеспечить одновременно правильность, доступность и устойчивость к разделению. Одновременно выполняться могут только лишь два свойства из трех.

Менее строгая модель BASE (Basically Avaliable, Soft state, Eventual consistency) заменяет ACID в рамках NoSQL [8].

— Базовая доступность (англ. Base avaliability) – На любой запрос будет получен ответ о успешном или неуспешном завершении.

— Неустойчивое состояние (англ. Soft state) – Состояние системы неустойчиво – может изменяться со временем. Причем изменения могут происходить без внешнего взаимодействия.

— Согласованность в конечном счёте (англ. Eventual consistency) – В конечном счете база данных будет правильной и согласованной, несмотря на то, что она может быть несогласованна в момент времени.

Структуры данных в нереляционных базах данных отличаются в сравнении с используемыми в реляционной модели. Далее будут описаны некоторые типы моделей данных используемые в NoSQL базах данных.

1.3.1 Нереляционные модели данных

В данном разделе будут рассмотрены разные типы нереляционных моделей данных. Согласно подходу к хранению данных нереляционные базы данных можно классифицировать по перечисленным моделям:

— Столбцовая (Колоночная) модель – Несмотря на то, что и колоночная и реляционная база основываются на понятии строк и колонок, для колоночного хранилища нет необходимости определять колонки. В сравнении с РСУБД, где данные хранятся в виде строк в таблице, колоночное хранилище предполагает хранение данных как набор колонок. Колонка может включать себя множество атрибутов, поэтому её могут называть «Широкой». Это позволяет оптимизировать выполнение сложных запросов. Основные причины использования колоночного хранилища:

  — в качестве распределенного хранилища;

  — для пакетной обработки больших объемов данных, предполагающих операции аггрегации, сортировки либо другой обработки данных;

  — для создания аналитических хранилищ.

  Множество популярных NoSQL СУБД используют колоночную модель данных, например: Apache Cassandra, HBase, Bigtable, DynamoDB.

— Документо-ориентированная модель – хранилище спроектированное для хранения данных в виде документов, которое позволяют записывать, получить и изменять слабо-структурированные данные. Для того, чтобы облегчить работу разработчика документы в документо-ориентированном хранилище не зависят от конкретной схемы. В документо-ориентированной базе данных, все данные представляются в виде документа в определенном формате. Документы могут быть в таких форматах как: Некоторые форматы документов: XML, JSON, BSON, YAML и другие. На рисунке 1.2 представлен документ в формате JSON. Также следует отметить, что документ может содержать другой документ внутри себя.

  В документ-ориентированных СУБД каждый документ обязан иметь уникальный идентификатор за его генерацию зачастую отвечает СУБД.

Рисунок 1.2 – Пример структуры данных в документо-ориентированной БД

В основном документ-ориентированная модель используется для создания web-приложний, имеющих потребность в обработке крупных объемов распределенных по сети текстовых данных. В качестве примеров популярных документо-ориентированных СУБД можно привести: MongoDB, CouchDB, Terrastore.

— Модель Ключ-Значение – в модели ключ-значение отсутствует схема и все данные хранятся в единой хэш-таблице. Идентификаторы или ключи символьные и могут быть как сгенерированными системой, так и заданными разработчиком вручную, аналогично идентификатору документа в документо-ориентированной базе данных. Хранилища типа ключ-значение в основном используются в виде ‘in-memory’ распределенного кэша, для организации быстрого доступа к данным. Среди наиболее популярных СУБД основанных на модели ключ-значение есть такие как Memcached, MemcacheDB и Redis.

— Графовая модель – Графовые базы данных хранят данные в виде графовых структур, состоящих из узлов, рёбер и свойств. Узлы отображают концептуальные объекты, соединенные линиями, которые называют

ребрами. Ребра используются для создания связей между узлами и свойствами. Как в реляционной модели, графовые базы данных работают с отношениями путем прохождения по ребрам. Используя графовые алгоритмы графовые базы данных могут хранить данные в таком виде, который можно масштабировать по нескольким серверам для дальнейших связей ребрами. Узлы и отношения – базовые понятия графовой модели, в которой узлы организуются по свойствам ассоциированным с отношениями, где данные хранимые в узлах также имеют собственные свойства.

Графовые базы данных обычно используются в случаях, когда отношения между данными более важны. Например, многие социальные сети используют графовые базы данных. Среди известных графовых СУБД можно выделить Neo4J, FlockDB, AllegroGraph.

1.3.2 Теорема Брюера (CAP)

В 2000г. Эриком Брюером была выдвинута CAP-теорема [9; 10]. Основная ее идея звучит как: “В любой реализации параллельных вычислений нельзя добиться одновременно не более двух свойств, таких как согласованность данных, доступность и устойчивость к разделению”. Каждой системе необходимо соответствовать следующим свойствам [11]:

— согласованность данных (англ. Consistency) – данные в каждом угле не должны противоречить друг другу и имеют общее согласованное состояние;

— доступность (англ. Avaliability) – любой запрос к распределенной системе завершается корректно;

— устойчивость к разделению (англ. Partition tolerance) – расщепление распределенной системы на изолированные сегменты не приводит к некорректности работы системы. Таким образом при отказе одного из узлов гарантируется работа общей системы.

Однако удовлетворить всем трем свойствам не предоставляется возможным. Согласованность данных легко достигается в реляционной базе данных, в силу соответствия ACID. В то же время, РСУБД проблематично горизонтально масштабировать, однако NoSQL базы данных просто масштабируются, но не могут предоставить тот же уровень согласованности данных, в силу соответствия BASE.

1.3.3 Сравнение NoSQL баз данных

На таблице 1.2 указаны результаты сравнения NoSQL СУБД разных моделей данных.

Таблица 1.2 – Сравнение наиболее популярных NoSQL СУБД.

Название	Модель данных	Выполняемые свойства CAP-теоремы	Модель целостности	Область применения
Redis	Ключ-значение	Согласованность Устойчивость к разделению	BASE	Кэширование данных сессий
MongoDB	Документо-ориентированная	Согласованность Устойчивость к разделению	BASE	Хранение больших коллекций данных
Cassandra	Колоночно-ориентированная	Доступность Устойчивость к разделению	BASE	Распределенное хранение и обработка больших данных
Neo4j	Графовая	Согласованность Доступность	ACID	Хранение и анализ социальных графов

1.3.4 Сравнительный анализ РСУБД и нереляционныхх СУБД

— Надежность транзакции: РСУБД соответствует ACID и предоставляет надежность транзакции. Нереляционные базы, опираются на ослабленную спецификацию — BASE.

— Модель данных: РСУБД основаны на реляционной модели, и данные представляются в виде таблиц с набором строк связанных отношением. Нереляционные базы данных используют различные модели, например: ключ-значение, документо-ориентированные, колоночные и графовые.

— Масштабируемость: Web-приложения требуют возможности горизонтального масштабирования, т. к. они распределены по нескольким серверам. NoSQL базы данных поддерживают горизонтальное масштабирование, в свою очередь горизонтальное масштабирование РСУБД является трудоемкой задачей.

— Обработка больших данных: Из за проблем с масштабированием и партицированием данных в распределенной кластеризованной среде, обработка больших объемов данных на РСУБД проблематична.

— Устойчивость к сбоям: Данные в РСУБД консистентны и поддерживаются в правильном состоянии за счет транзакций и журналирования. Сохранность данных в нереляционных базах зависит от правильности их репликации.

— Безопасность: В РСУБД внедряются сложные механизмы безопасности. Нереляционные базы спроектированы таким образом, чтобы добиться производительности в обработке больших объемов данных, засчет потерь в их безопасности. Безопасность данных является крупным объектом обсуждения в рамках активно развивающихся облачных систем.

В таблице 1.3 приведено сравнение реляционных и нереляционных СУБД.

Таблица 1.3 – Сравнение РСУБД и NoSQL СУБД

Вид / Критерий	РСУБД	Нереляционные СУБД
Надежность транзакции	Да (ACID)	Нет (BASE)
Модель данных	Реляционная	Множество нереляционных
Горизонтальное масштабирование	Нет	Да
Обработка больших данных	Не предназначались, имеют ограничения	Учитывают потребности в обработке больших данных
Надежность	Да	Зависит от репликации
Безопасность	Да	Нет

1.4 Миграция данных

Миграция – процесс переноса данных из одной или более текущих баз дан- ных в одну или более новых баз данных с возможным изменением структуры данных с помощью использования некоторого инструмента для миграций.

Промышленные применения реляционной модели данных не поддерживают растущие требования по производительности в сравнении с нереляционными в рамках анализа больших объемов данных [12]. Миграция данных это необходимая часть промышленного статистического анализа данных. Проведя сравнительный анализ РСУБД и нереляционных СУБД, можно сделать вывод о том, что основное их различие состоит в модели данных и том в виде какой структуры данные хранятся.

1.5 Выбор баз данных

Для более детального рассмотрения был выбран процесс миграции данных с РСУБД на документо-ориентированную СУБД. Выбор документо-ориентированной СУБД обусловлен тем, что документо-ориентированные СУБД являются вторыми по распространенности после реляционных [1]. Так-

же стоит учесть, что принципы документо-ориентированной модели явно противоречат реляционной, так как оптимальным подходом является денормализация схемы, в противовес нормализации. Помимо этого открытыми являются проблемы избыточности данных после миграции и переноса отношений «многие-к-многим».

1.6 Описание процесса миграции

С учетом структуры и деталей хранения, нереляционные СУБД отличаются от РСУБД. Реляционная модель строго структурирована и данные нормализованы в таблицах согласно их отношениям, когда в нереляционных базах данных данные хранятся в полуструктурированном или неструктурированном виде и они обычно денормализованы. Таким образом процесс миграции не является тривиальной задачей.

На рис 1.3 представлена общая схема процесса миграции.

Рисунок 1.3 – Схема процесса миграции

1.7 Методы миграции

Существуют разные подходы к миграции данных. Они в основном предлагают различную методологию по переносу данных и схемы из реляционного хранилища в документо-ориентированное. Основной задачей, при переносе схемы и данных является представление реляционных отношений. Методы могут быть основаны как на анализе и преобразовании схемы данных, так и не зависеть от схемы и манипулировать только данными.

Далее будут рассмотрены несколько методов миграции.

1.7.1 Методы на основе аггрегации

Одни из наиболее распространенных подходов при миграции в документо-ориентированное хранилище из РСУБД основаны на аггрегации. В основе метода лежит идея о денормализации структуры, путем аггрегирования всех зависимых сущностей в единую главную сущность. Таким образом из множества таблиц и отношений между ними образуется единый документ, где все зависимые документы являются вложенными внутрь основного документа. Подобные подходы позволяют получать тесно связанные данные одним запросом к одной большой сущности, таким образом получается избежать JOIN-операций, не поддерживаемых в множестве нереляционных СУБД, однако миграция сложных схем подобным образом приводит к большой избыточности данных и их повторению. Также стоит отметить, что в случае повторения данных необходимо самостоятельно следить за их согласованностью.

В статье [13] представлен полу-автоматизированный метод преобразования реляционной базы данных в документо-ориентированную на основе табличной денормализации. Пользователю предлагается выбрать необходимые для переноса главную и зависимые сущности. Далее все зависимые сущности переносятся в виде вложенных в главный документов. Из недостатков этого метода можно выделить то, что он не позволяет сразу перенести всю реляционную структуру и требует вмешательства пользователя. Также при излишней денормализации растет объем документа и как следствие выполнение сложных запросов с несколькими коллекциями становится менее эффективным. Помимо этого такой подход неприменим к сложным реляционным структурам, так как конверсия в единый денормализованный документ не всегда возможна.

На рисунке 1.4 представлена общая схема алгоритма на основе табличной денормализации.

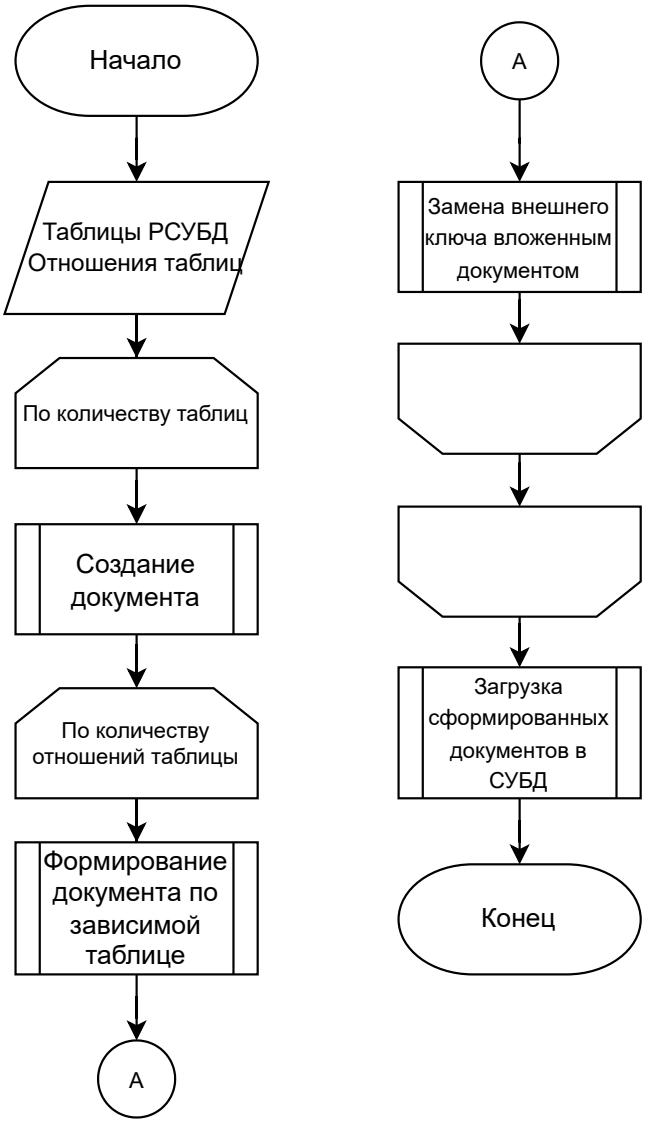

Рисунок 1.4 – Схема алгоритма на основе табличной денормализации

Также существуют полностью автоматизированные методы денормализации. В статье [14] представлен метод миграции схемы основанный на анализе графа отношений. Предлагается рассматривать реляционную схему как направленный ациклический граф. Далее на основе понятия полноты данных вводятся операции расширения узла. Так как изначально каждый узел представляет собой сущность или таблицу в результате работы алгоритма граф преобразуется таким образом, что в каждом узле хранится вся зависимая от него информация.

На рисунке 1.5 приведена схема алгоритма для поиска цепочки всех узлов необходимых для дополнения сущности. В качестве входных данных принимаются значения:

$$O[v] \leftarrow |b(v)|,$$

$$P \leftarrow \{v | v \in V \wedge O[v] = 0\},$$

$$Q \leftarrow V - P,$$

$$T \leftarrow \emptyset,$$

$$S \leftarrow \{\},$$

где:

— $O[v]$ – количество входящих граней;

— Q – узлы у который $O[v] = 0$;

— P – узлы имеющие входящие грани и не имеющие исходящих;

— T – узлы не имеющие ни входящих не исходящих граней;

— S – путь по графу, составляющий полную сущность;

— $f(x)$ – узлы, имеющие прямой путь в x.

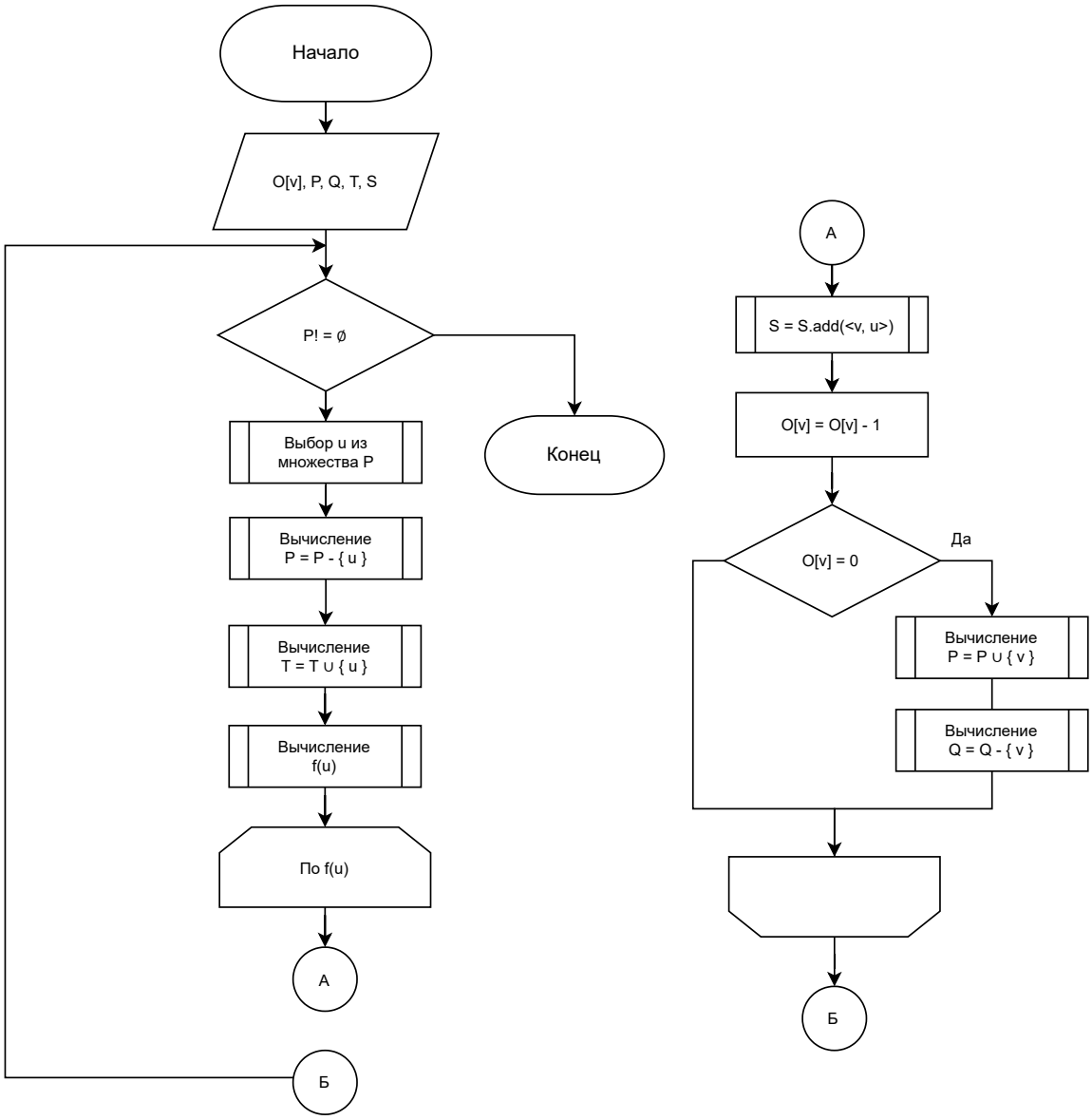

Рисунок 1.5 – Схема алгоритма на основе рашсширения узлов направленного ациклического графа

В итоге получаем несколько денормализованных коллекций документов, с которыми можно работать не прибегая к JOIN-операциям. Однако такой подход ведет к высокой избыточности, так как все зависимые данные будут дублироваться. Также к недостаткам метода можно отнести то, что не все графы отношений можно выразить с помощью направленного ациклического графа. Таким образом подобный метод не подходит для переноса схем имеющих отношения «многие-к-многим».

1.7.2 Методы на основе нормализации

Несмотря на то, что документо-ориентированная схема предполагает денормализацию данных, существуют методы напрямую переносящие нормализованную реляционную схему в документо-ориентированную модель. В таком случае отношения реализуются не засчет вложенности документов, а с помощью внешних ключей. Внешние ключи определяются так же как и в РСУБД – через указание первичного ключа (идентификатора) другой сущности (документа).

В статье [15] описывается каким образом нормализованные данные из РСУБД могут быть представлены в документо-ориентированной СУБД без нарушения нормальной формы. На основе этого может быть построен алгоритм согласно которому все данные из РСУБД переносятся с сохранением всех своих внешних и первичных ключей.

Однако, в силу того, что документо-ориентированные СУБД не подразумевают нормализованное представление данных, сохранение согласованности данных и реализация JOIN-операций отводится разработчику. Это значительно затрудняет операции вставки, изменения и удаления, так как СУБД не берет на себя отслеживание изменений в зависимых сущностях.

На рисунке 1.6 представлена схема алгоритма переноса нормализованной схемы.

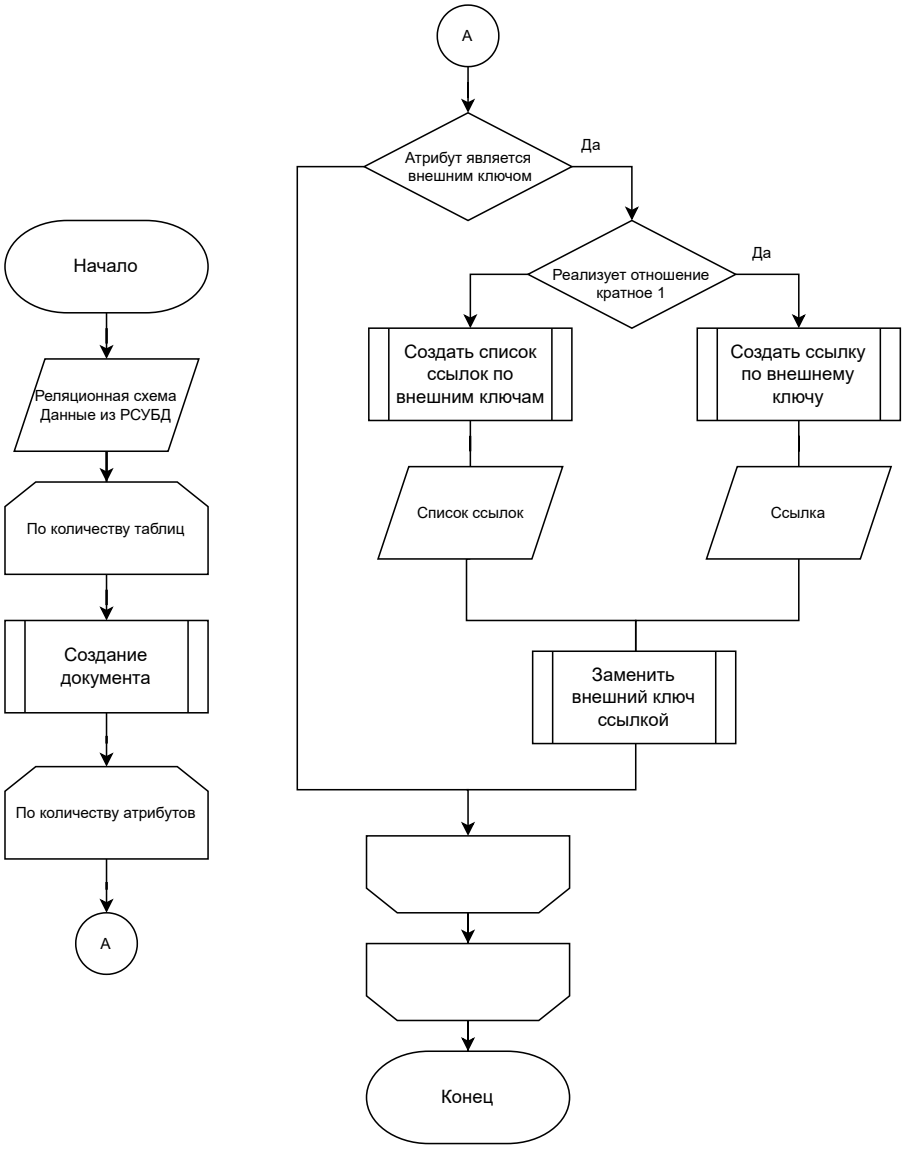

Рисунок 1.6 – Схема алгоритма переноса нормализованной схемы с использованием ссылок

В статье [16] предлагается подход на основе перевода таблицы из РСУБД в формат CSV (Comma Separated Value) и далее загрузке данного файла в документо-ориентированную СУБД. Данный метод очень сильно зависит от того, каким образом документо-ориентированная СУБД будет воспринимать полученный формат файла. Зачастую при использовании подобного подхода могут не учитываться отношения реляционной модели и тем самым мы теряем возможность обращения к связанным между собой данным. Поэтому данный подход предпочтителен, когда необходимо перенести конкретную таблицу из реляционной в документо-ориентированную базу данных. На рисунке 1.7 представлена общая схема метода миграции через CSV файл.

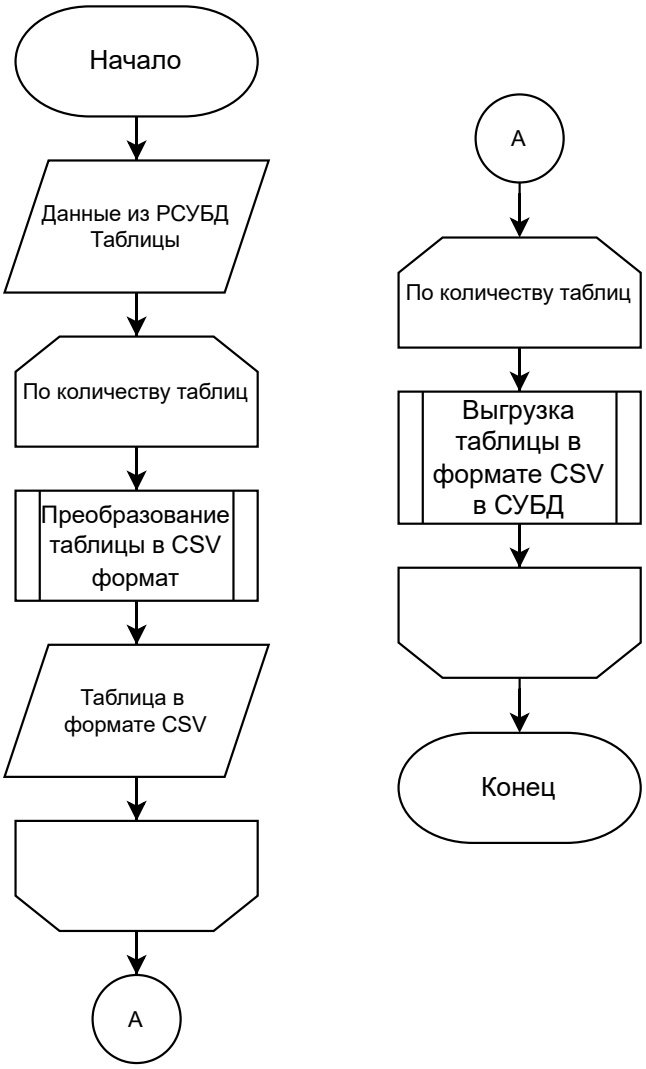

Рисунок 1.7 – Схема алгоритма на основе перевода в CSV файл

1.7.3 Метод основанный на введении нереляционной схемы

Методы вводящие понятие собственной нереляционной схемы частично объединяют описанные выше подходы. Зачастую предлагается введение методологии или набора правил, согласно которым реляционная схема приводится к виду совместимому с документо-ориентированной моделью. В статье [17] описывается один из таких методов. Предлагается набор правил, согласно которому на основе вида сущности и отношения принимается решение о том, каким образом реализуется данное отношение в документо-ориентированной модели. Отношение может быть реализовано как с помощью ссылки, так и зачет вложенности документов.

Метод предлагает подход для миграции данных в таком формате, который является компромиссным между нормализованным и денормализованным. Также данный метод предлагает решение для переноса связей «многие-к-многим» хоть и не решает проблем с отслеживанием согласованности данных в рамках этого отношения. Засчет этого он может использоваться для переноса сложных реляционных структур, в то время как методы на основе аггрегации зачастую не применимы к структурам, которые нельзя представить в виде направленного ациклического графа.

Таким образом предлагается оптимальное решение не приводящее к излишней избыточности, при этом более оптимальное с точки зрения хранения данных и выполнения запросов нежели перенос нормализованной структуры с сохранением нормальной формы.

На рисунке 1.8 представлена общая схема алгоритма преобразования реляционной схемы в документо-ориентированную согласно методологии описанной в [17];

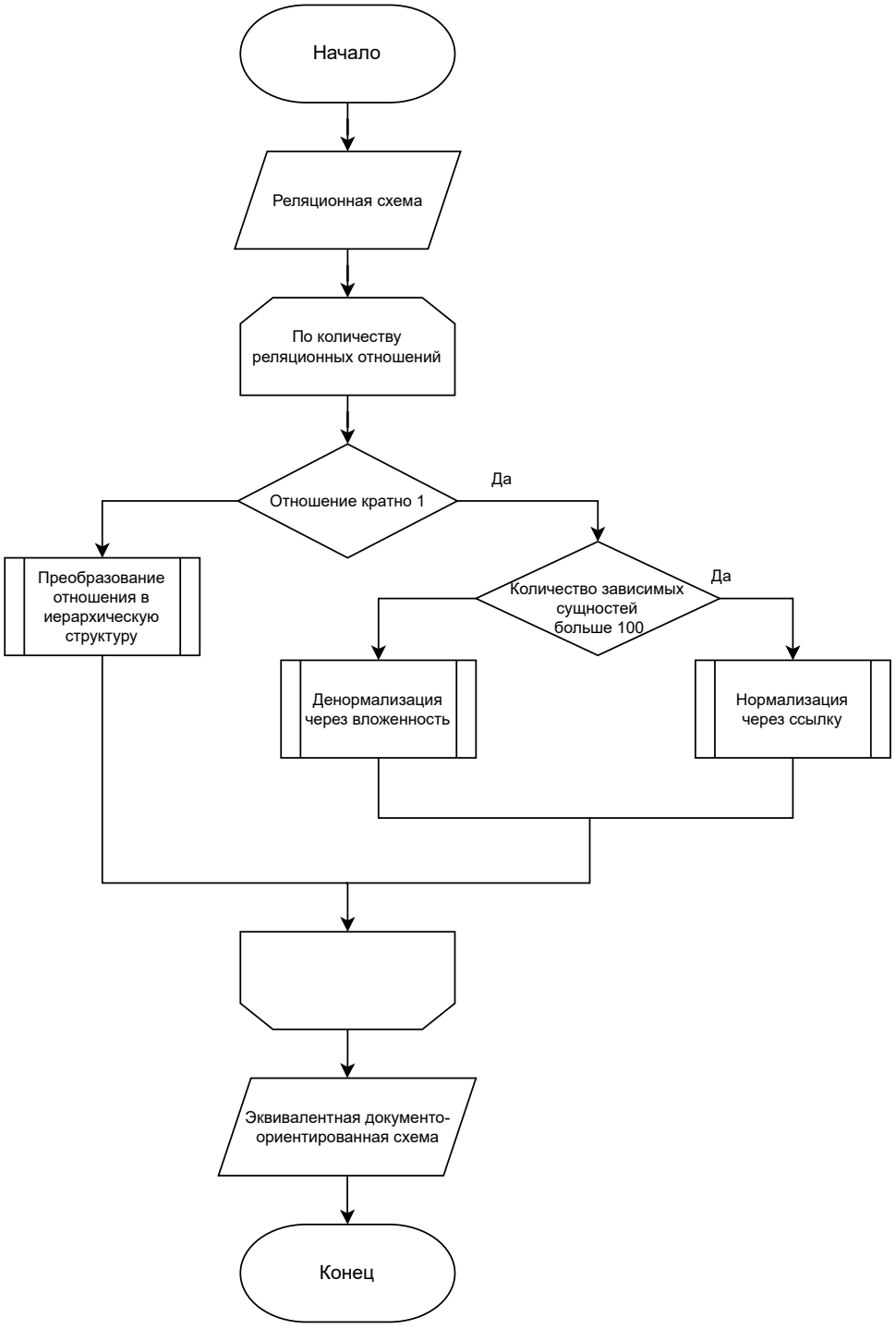

Рисунок 1.8 – Схема алгоритма на основе нереляционной схемы DODS

1.7.4 Метод использующий оберточный слой для запросов

Методы использующие оберточный слой для запросов позволяют проводить процесс миграции без анализа схемы исходной реляционной базы данных. Сравнение конкретных систем использующих этот метод представлено в статье [18]. Подход с оберточным слоем предполагает создание слоя преобразующего запросы на языке SQL в запросы совместимые с желаемой нереляционной СУБД. Таким образом данные миграция данных осуществляется засчет средств языка SQL, который поддерживается большинством реляционных СУБД. Такой подход позволяет объединить приемущества хранения в документо-ориентированной модели и возможности языка запросов SQL, однако результирующая структура данных может быть не оптимальной в рамках нереляционной модели данных.

На рисунке 1.9 представлена общая схема метода использующего оберточный слой для запросов.

Рисунок 1.9 – Схема метода использующего оберточный слой для запросов

1.7.5 Методы на основе объектного преобразования

Существуют методы на основе объектных преобразований, они используют объектно-реляционные (ORM) и объектно-нереляционные (ONM) преобразования. Использование подобного рода систем позволяет ввести промежуточное представление сущности в виде объекта, что помогает создать единообразный интерфейс взаимодействия с базами данных не зависимо от их модели, а также облегчает разработчикам взаимодействие с данными в рамках объектно-ориентированной парадигмы. В работе [19] рассмотрено несколько подобных систем.

Например система Hibernate широко используется для работы и с реляционными и нереляционными базами данных и предлагает собственные языки запросов HQL и JPQL, что позволяет единообразно обращаться к многим видам баз данных [20]. Такие системы получили популярность так как облегчают наиболее распространенные операции получения и изменения данных, однако они не являются оптимальными для работы с большими приложениями, где требуется множество сложных аналитических запросов к данным. Аналогичный подход с использованием системы LinQ рассмотрен в статье [21].

На рисунке 1.10 представлена общая схема метода миграции на основе объектного преобразования.

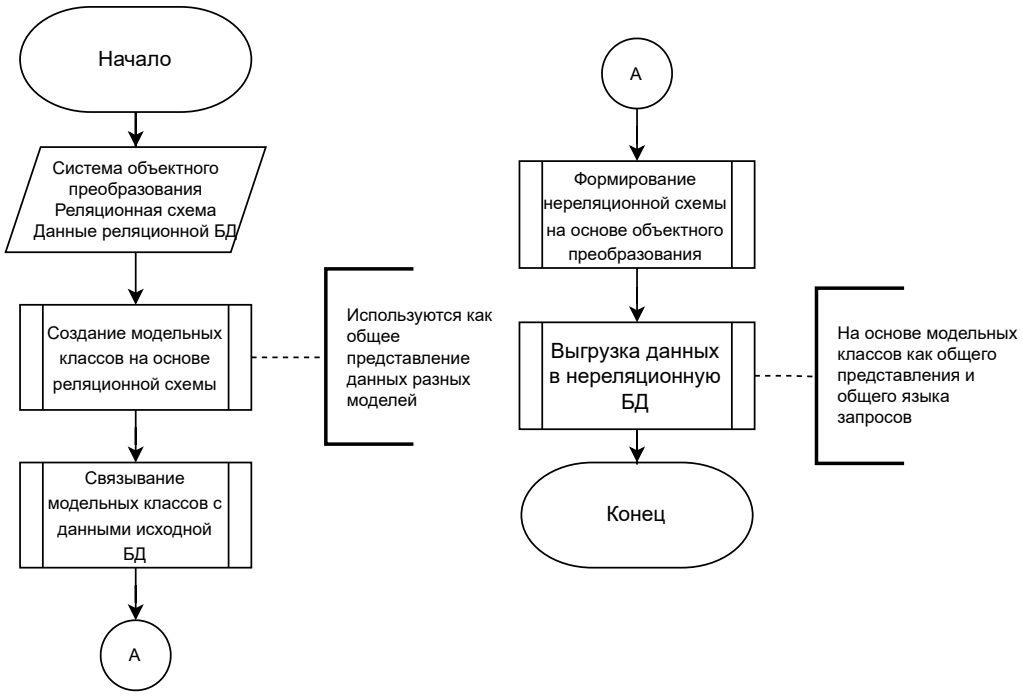

Рисунок 1.10 – Схема алгоритма на основе объектного преобразования

1.7.6 Методы на основе ETL

ETL (Extract Transform Load) – подход к обработке данных, преполагающий последовательное получение данных, преобразование и выгрузку. На основе этого подхода основаны некоторые методы миграции данных. В статье [22] предложен метод миграции данных из реляционной БД в документо-ориентированную основанный на ETL подходе. Сначала путём подключения к реляционной БД получается структура таблиц. Далее пользователю предоставляется возможность выбрать данные необходимые для переноса в новую БД. После выбора необходимых таблиц и аттрибутов, происходит выгрузка данных и преобразование их в формат совместимый с документо-ориентированной моделью, затем данные загружаются в новую базу данных. Метод нельзя назвать полностью автоматизированным, так как этап определения новой схемы требует вмешательства пользователя, однако этот метод автоматизирует этап переноса данных в выбранную пользователем схему.

Существуют решения с открытым исходным кодом использующим данный подход, например: MongoSyphon [23].

На рисунке 1.11 представлена общая схема ETL подхода.

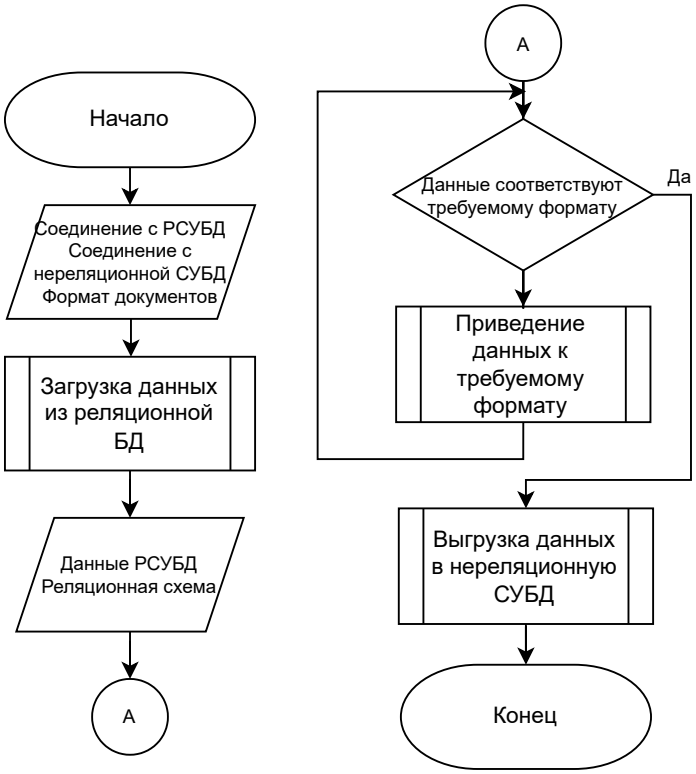

Рисунок 1.11 – Схема алгоритма ETL

1.8 Сравнение методов миграции

В ходе исследования были установлены следующие критерии сравнения методов миграции:

— применимость – применимость метода включает в себя два критерия: применимость к схемам и применимость к сущностям.

Применимость к схемам показывает, к каким схемам наиболее применим данный метод. Схема называется сложной если она включает в себя более 10 сущностей.

Применимость к сущностям показывает, к миграции сущностей какого размера применим данный метод. Сущность называется большой если она насчитывает более 10 атрибутов и имеет более двух отношний.

— поддержка отношения «многие-к-многим» – данный критерий показывает, позволяет ли метод переносить отношения типа «многие-к-многим» из РСУБД в документо-ориентированную;

— избыточность данных – данный критерий определяет количество дублирующейся информации при миграции. Избыточность определяется как высокая, если все зависимые данные дублируются для каждой сущности или как низкая, если дублирование данных не присутствует. Средняя избыточность показывает, что данные дублируются, но в количестве меньшем количества зависимых сущностей;

— автономность – данный критерий показывает потребность вмешательства разработчика в процесс миграции.

1.8.1 Сводная таблица

В таблице 1.4 приведено сравнение методов миграции по сформулированным критериям.

Таблица 1.4 – Сравнение методов миграции

Критерий / Название	Применимость		Избыточность данных	Поддержка отношения «многие-к-многим»	Автономность
	К схеме	К сущностям			
Метод на основе табличной денормализации	Простые	Малые	Максимальная	Нет	Нет
Графовый метод на основе аггрегации	Сложные	Малые	Максимальная	Нет	Да
Метод на основе нормализации	Сложные	Большие	Минимальная	Да	Да
Метод на основе нереляционной схемы	Любые	Любые	Средняя	Да	Да
Метод на основе обертки запросов к СУБД	Любые	Любые	Средняя (Зависит от реализации)	Да	Нет
Метод на основе объектного преобразования	Простые	Малые	Средняя	Да	Нет
Метод на основе ETL процесса	Любые	Большие	Средняя (Зависит от реализации)	Да	Нет

Вывод

Сравнение показало, что метод на основе введения нереляционной схемы является оптимальным, так как он имеет возможность полной автоматизации и предполагает возможность оптимизации количества дубликатов данных. Также методология предлагаемая данным методом имеет возможность быть модифицированной на основе дополнительных входных данных.

1.9 Применение миграции данных

Задача миграции данных из реляционного в документо-ориентированное хранилище возникает по разным причинам:

1. Масштабирование приложения — при росте Web-приложений возникает потребность в увеличении объема и производительности хранилища, зачастую это достигается увеличением количества серверов и вынесением приложения в распределенную среду. Таким образом возникает потребность в переносе данных в более приспособленное для таких условий хранилище.

2. Отказ от строгой структуры данных — иногда оказывается, что спроектированная в ходе разработки реляционная схема не подходит для решаемой задачи. В таком случае возникает потребность в переносе данных в хранилище с менее строгой схемой, например документо-ориентированное.

3. Аналитика — накопление больших объемов данных затрудняет работу с РСУБД. В таких случаях прибегают к созданию нереляционных хранилищ, на которых проще и эффективнее проводить аналитику.

1.10 Постановка задачи миграции из реляционного в документо-ориентированное хранилище

Задача миграции данных состоит в том, чтобы перенести данные из одной базы данных в другую наиболее оптимальным образом. В силу того, что реляционная и документо-ориентированная модели значительно отличаются такой процесс не является тривиальным. Возникают проблемы в переносе отношений и организации доступа к данным в новой модели. В ходе исследования было замечено, что большая часть методов не учитывает особенности взаимодействия с ранее хранимыми данными. Как следствие, в качестве метода решения задачи был выбран метод на основе введения нереляционной схемы. Однако, рассмотренные в ходе работы методы никак не учитывают статистику запросов к РСУБД. Таким бразом предлагается оптимизация процесса миграции на основе статистики запросов. Например, в ходе миграции стоит учесть, что данные которые чаще запрашиваются совместно, стоит аггрегировать.

На рисунке 1.12 представлена постановка задачи в форме IDEF0- диаграммы.

Рисунок 1.12 – IDEF0-диаграмма процесса миграции

2 Конструкторский раздел

В данном разделе представлены результаты разработки метода миграции данных из реляционной в документо-ориентированную базу данных с использованием частотного и семантического анализа. Представлены используемые алгоритмы, описаны входные и выходные данные. Описан процесс преобразования схемы из реляционной в документо-ориентированную. Выполнено проектирование для программной реализации метода с использованием функциональной модели IDEF0 и схем алгоритмов.

2.1 Описание метода миграции данных

Входными данными являются:

1. Реляционная схема.
2. Статистика запросов к СУБД.
3. Информация о предметной области.
4. Данные реляционной БД.

В ходе работы метода происходит сбор метаданных о таблицах РСУБД, таких как названия таблиц, первичные и внешние ключи, столбцы и типы данных. Далее эти метаданные дополняются результатами частотного анализа JOIN-операций, засчет того, что связи между сущностями помечаются как «часто объединяемые». На основе сформированных метаданных, засчет семантического анализа происходит формирование JSON-схемы, как промежуточного представления данных в документо-ориентированной БД. Согласно сформированной схеме, производится перенос данных из реляционной в документо-ориентированную БД.

Выходными данныи являются:

1. Данные в документо-ориентированной БД.

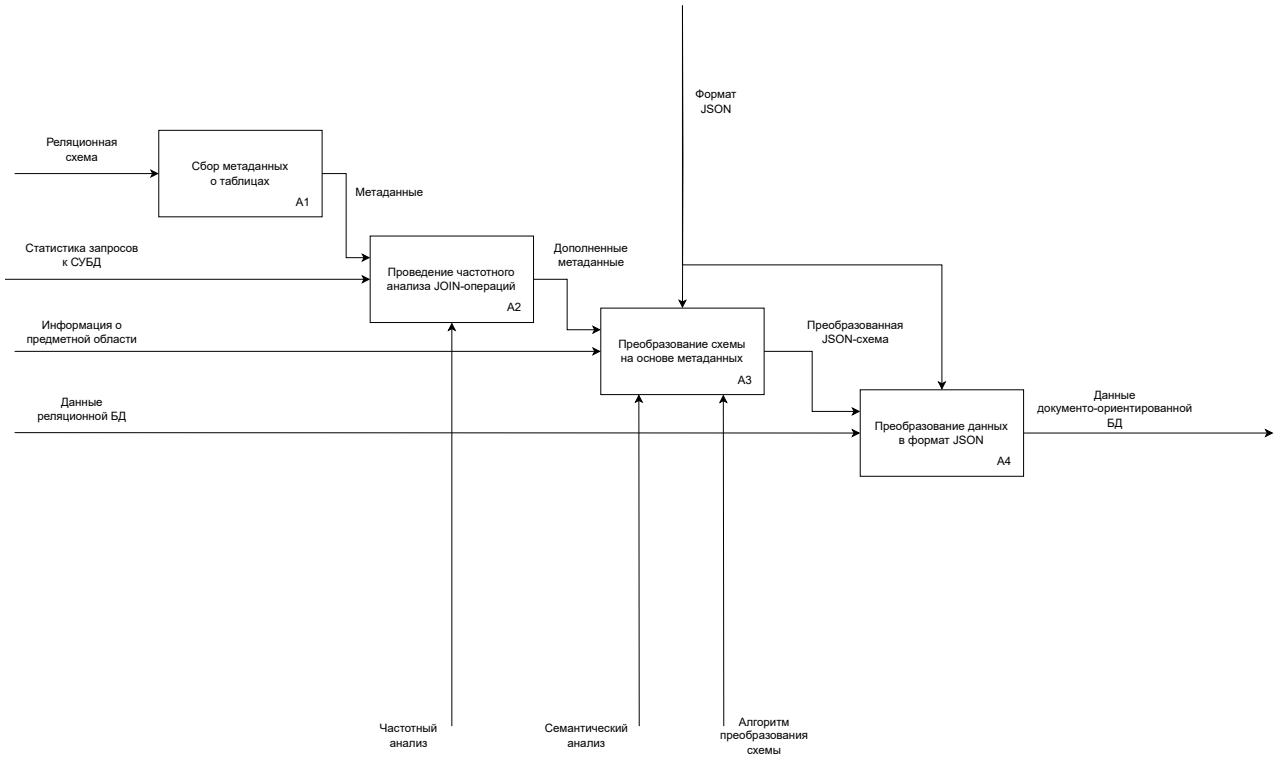

Рисунок 2.1 – Подпись

2.2 Описание алгоритма получения метаданных о реляционной схеме

Реляционные СУБД хранят информацию о созданных в них таблицах в отдельной схеме, которая называется information_schema и является частью стандарта ANSI [?]. Данные о таблицах, их столбцах и ключах хранятся в виде информационных таблиц и представлений, что позволяет поддержание стандарта SQL, тем самым позволяет использовать предлагаемый инструментарий в большинстве РСУБД. Многие СУБД также предоставляют собственные дополнительные инструменты для получения метаданных, например, системные каталоги, но они прориетарны для каждой системы управления базами данных и не портируемы.

В первую очередь необходимо собрать сниок интересующих нас таблиц. Далее, для каждой из этих таблиц нужно получить информацию о названиях столбцов, типах данных и ключах. Информация о ключах и связях между таблицами является наиболее важной, поэтому необходимо получить не только список первичных и внешних ключей в каждой таблице, но и внешние ключи, которые ссылаются на интересующие нас таблицы. Таким образом мы

получаем не только исходящие но и входящие связи.

В результате сбора метаданных мы получаем список структур, каждая из которых хранит метаданные для одной таблицы.
Структуры имеют вид:

— название таблицы;

— список столбцов;

— список первичных ключей;

— список внешних ключей;

— список исходящих связей таблицы;

— список входящих связей таблицы;

Собранные метаданные служат основой для семантического анализа схемы, который позволяет предварительно определить отношения между таблицами для дальнейшего выбора метода переноса связи из реляционной в документо-ориентированную модель.

Тем не менее, метаданные не содержат достаточно информации для организации процесса миграции, так как информация о кратности отношений и интересущих нас таблицах зависит от особенностей предметной области и не может быть определена без данных от пользователя.

На рисунке 2.2 представлен алгоритм сбора метаданных о таблицах базы данных.

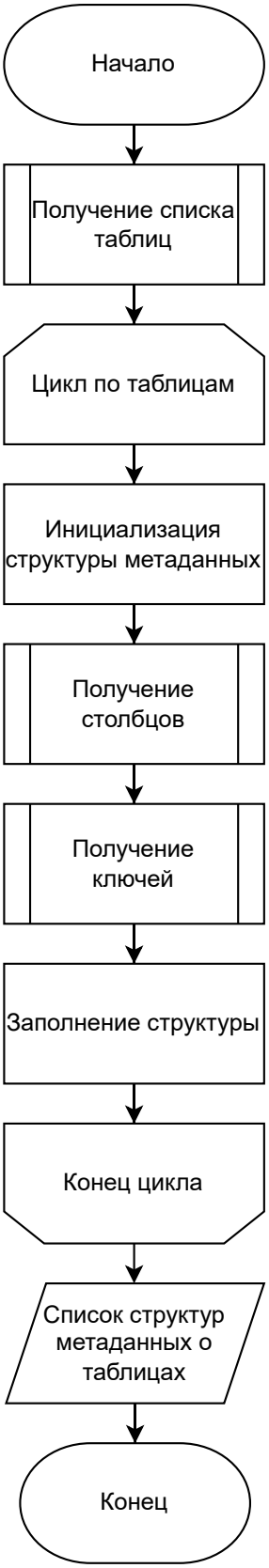

Рисунок 2.2 – Алгоритм сбора метаданных таблиц реляционной базы данных.

2.3 Описание алгоритма частотного анализа JOIN-операций

Большинство современных РСУБД предоставляют возможность логирования и сбора статистики о работе СУБД. Этот инструментарий может быть использован для анализа запросов к базе данных.

История запросов к СУБД обрабатывается, для определения частоты совершения JOIN-операций между двумя таблицами. Если таблица участвует в JOIN-операции значительно чаще чем в SELECT, можно предположить, что данные в зависимой таблице не востребованны отдельно от основной, поэтому она выдвигается как кандидиат для пометки как «часто объединяемая».

На основе этх данных, «часто объединяемые» таблицы рекомендуются к предаггрегации. Таким образом в документо-ориентированной модели их отношение будет реализовано засчет вложенных документов.

На рисунке 2.3 представлен алгоритм дополнения метаданных результатами частотного анализа.

Рисунок 2.3 – Алгоритм дополнения метаданных результатами частотного анализа.

На рисунке 2.4 представлен алгоритм анализа истории запросов к базе данных.

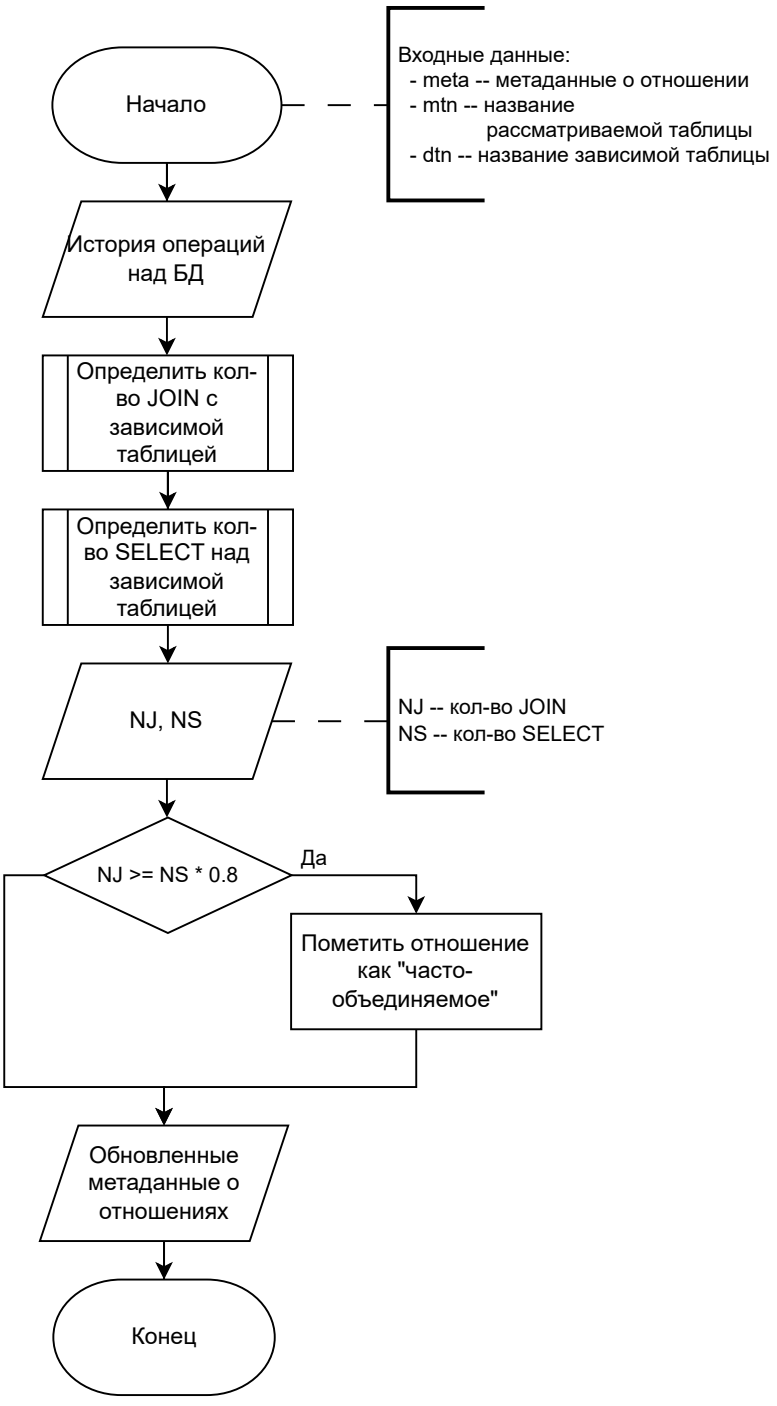

Рисунок 2.4 – Алгоритм анализа истории запросов.

2.4 Описание алгоритма преобразования схемы

После сбора и предобработки метаданных следует преобразовать реляционную схему в документо-ориентированную JSON-схему. Происходит выбор таблиц, на основе которых будут созданы коллекции документов в документо-ориентированной СУБД. Далее для каждой таблицы, на основе собранных метаданных с использованием информации о предметной области и семантического анализа формируется JSON-документ, который будет отражать вид данных в документо-ориентированной БД. Все неключевые столбцы таблицы преобразуются в JSON-атрибуты. Первичный ключ заменяется идентификатором объекта. После чего для каждого внешнего ключа предлагается набор действий в зависимости от имеющейся информации. Зависимые таблицы либо встраиваются в основной документ, либо создается ссылка на идентификатор из другой коллекции.

На рисунке 2.5 представлен алгоритм преобразования реляционной схемы в JSON-схему.

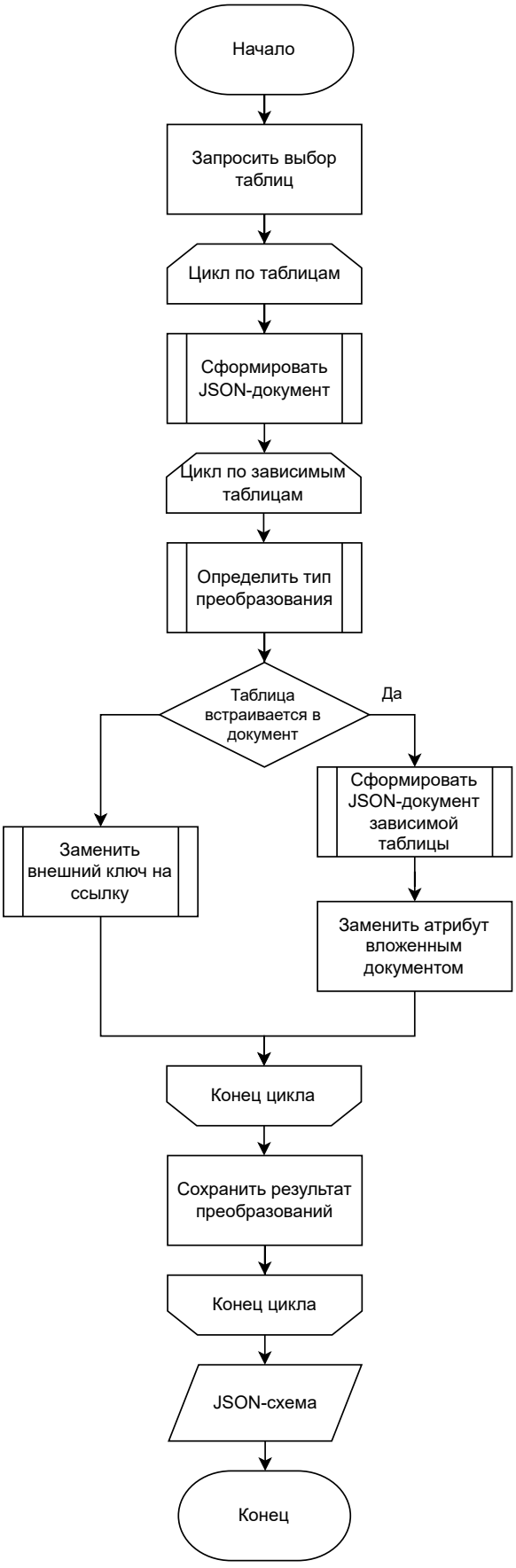

Рисунок 2.5 – Алгоритм преобразования схемы.

На рисунке 2.6 представлен алгоритм определения типа преобразования.

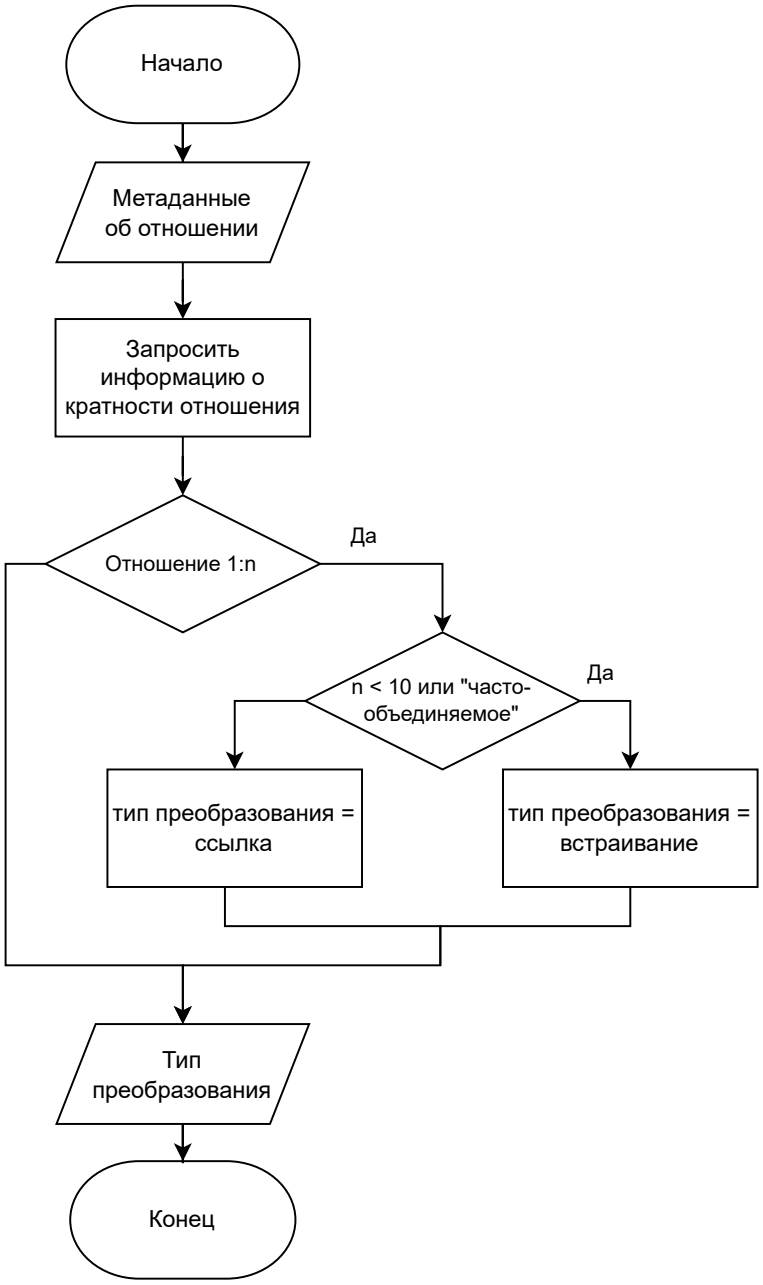

Рисунок 2.6 – Алгоритм определения типа преобразования.

2.5 Алгоритм преобразования данных в формат JSON

Сформированная схема используется для последующего преобразования данных. Согласно метаданным и сформированной схеме, данные реляционной БД выгружаются и каждая запись преобразуется в эквивалентный JSON-документ согласно схеме, после чего сформированный документ выгружается в целевую документо-ориентированную БД.

Рисунок 2.7 – Алгоритм преобразования данных.

Выводы

В данном разделе были описаны результаты разработки метода миграции данных из реляционной в документо-ориентированную базу данных с использованием семантического и частотного анализа. Были представлены используемые алгоритмы, описаны входные и выходные данные. Выполнено проектирование для дальнейшей программной реализации метода при помощи функциональной модели IDEF0 и схем алгоритмов.

3 Технологический раздел

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. DB-Engines Ranking [Электронный ресурс]. — Режим доступа URL: `https://db-engines.com/en/ranking` (Дата обращения: 10.11.2022).
2. *Codd E. F.* A Relational Model of Data for Large Shared Data Banks // Commun. ACM. — New York, NY, USA, 1970. — Янв. — Т. 13, № 6. — С. 377—387.
3. *Chen, Peter, Pin-Shan.* The Entity-Relationship Model—toward a Unified View of Data // ACM Trans. Database Syst. — New York, NY, USA, 1976. — Март. — Т. 1, № 1. — С. 9—36.
4. *Codd E. F.* Extending the Database Relational Model to Capture More Meaning // ACM Trans. Database Syst. — New York, NY, USA, 1979. — Дек. — Т. 4, № 4. — С. 397—434.
5. *Chamberlin D. D., Boyce R. F.* SEQUEL: A Structured English Query Language // Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control. — Ann Arbor, Michigan : Association for Computing Machinery, 1974. — С. 249—264.
6. *Date C.* An Introduction to Database Systems. — Addison-Wesley Publishing Company, 1995.
7. *Khan W., Kumar T., Cheng Z.* SQL and NoSQL Databases Software architectures performance analysis and assessments–A Systematic Literature review. — 2022.
8. *Sadalage P. J., Fowler M.* NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence //. — Addison-Wesley Professional, 2012.
9. *Brewer E. A.* Towards Robust Distributed Systems (Abstract) // Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing. — Portland, Oregon, USA : Association for Computing Machinery, 2000. — С. 7.
10. *Brewer E. A.* A Certain Freedom: Thoughts on the CAP Theorem // Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing. — Zurich, Switzerland : Association for Computing Machinery, 2010. — С. 335.

11. *Lee E. A.*, *Bateni S.*, *Lin S.* Quantifying and Generalizing the CAP Theorem. — 2021.

12. *Bansal N.*, *Soni K.*, *Sachdeva S.* Journey of Database Migration from RDBMS to NoSQL Data Stores // Big-Data-Analytics in Astronomy, Science, and Engineering. — Cham : Springer International Publishing, 2022. — С. 159—177.

13. *Arora R.*, *Aggarwal R. R.* An algorithm for transformation of data from MySQL to NoSQL (MongoDB) // International Journal of Advanced Studies in Computer Science and Engineering. — 2013. — Т. 2, № 1. — С. 6—12.

14. *Zhao G.*, *Lin Q.*, *Li L.* Schema Conversion Model of SQL Database to NoSQL // 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. — 2014. — С. 355—362.

15. *Gu Y.*, *Shen S.*, *Wang J.* Application of nosql database mongodb // 2015 IEEE International Conference on Consumer Electronics-Taiwan. — IEEE. 2015. — С. 158—159.

16. *Chickerur S.*, *Goudar A.*, *Kinnerkar A.* Comparison of relational database with document-oriented database (mongodb) for big data applications // 2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA). — IEEE. 2015. — С. 41—47.

17. *Hamouda S.*, *Zainol Z.* Document-Oriented Data Schema for Relational Database Migration to NoSQL // 2017 International Conference on Big Data Innovations and Applications (Innovate-Data). — 2017. — С. 43—50.

18. *Schreiner G. A.*, *Duarte D.*, *Mello R. d. S.* When relational-based applications go to NoSQL databases: A survey // Information. — 2019. — Т. 10, № 7. — С. 241.

19. *Störl U.*, *Hauf T.*, *Klettke M.* Schemaless NoSQL data stores-Object-NoSQL Mappers to the rescue? // Datenbanksysteme für Business, Technologie und Web (BTW 2015). — 2015.

20. Hibernate OGM [Электронный ресурс]. — Режим доступа URL: `https://hibernate.org/ogm/` (Дата обращения: 16.11.2022).

21. *Drenckberg S. G.*, *Politze M.*, *Center I.* Migration of a web service back-end from a relational to a document-oriented database. — 2018.

22. *Hanine M.*, *Bendarag A.*, *Boutkhoum O.* Data migration methodology from relational to NoSQL databases // International Journal of Computer, Electrical, Automation, Control and Information Engineering. — 2015. — Т. 9, № 12. — С. 2566—2570.

23. MongoSyphon [Электронный ресурс]. — Режим доступа URL: `https://github.com/johnlpage/MongoSyphon` (Дата обращения: 26.11.2022).

ПРИЛОЖЕНИЕ А