Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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 автоматической нормализации
для реляционных баз данных
с использованием анализа функциональных зависимостей

Студент ИУ7-85Б _____ Т. А. Зуев
(группа) (подпись) (инициалы, фамилия)

Руководитель ВКР _____ А. Л. Исаев
(подпись) (инициалы, фамилия)

Нормоконтролер _____ _____
(подпись) (инициалы, фамилия)

2025 год

РЕФЕРАТ

Расчетно-пояснительная записка содержит 60 страниц, 20 рисунков, 34 источника.

Ключевые слова: реляционная модель, декомпозиция отношений, нормализация.

Задачи, решаемые в работе:

— Анализ предметной области реляционных баз данных;

— Проектирование метода автоматической нормализации;

— Разработка метода автоматической нормализации;

— Исследование временных характеристик метода в зависимости от сложности входных данных.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

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

— База данных — это организованная совокупность данных, хранимая и управляемая с помощью специализированной системы управления базами данных (СУБД), предназначенная для эффективного хранения, поиска, обработки и модификации данных. База данных представляет собой структурированное хранилище информации, где данные организованы таким образом, чтобы обеспечивать их целостность, согласованность и доступность для пользователей и приложений.

— НФ — нормальная форма.

— СУБД — средство управления базами данных.

ВВЕДЕНИЕ

Согласно исследованием, мировые объемы хранимых данных ежегодно растут, при этом скорость роста также ежегодно возрастает [1]. В связи с этим задача эффективного проектирования баз данных приобретает первостепенное значение. Особую актуальность эта задача имеет для реляционных баз данных, которые, по оценкам на 2021 год, составляют около 80% всего рынка баз данных [2]. Ключевым аспектом проектирования, направленным на обеспечение целостности, непротиворечивости и минимизацию объема хранимых данных, является нормализация [3]. Теоретическим фундаментом этого процесса служит математический аппарат функциональных зависимостей, описывающий семантические связи между атрибутами в реляционной модели. Традиционно процедура нормализации выполняется вручную экспертами-проектировщиками, что сопряжено с рядом существенных недостатков: высокие требования к квалификации специалиста, значительные трудозатраты и высокая вероятность допущения ошибок, особенно в схемах с большим количеством атрибутов и сложными зависимостями [4]. Эти факторы существенно ограничивают масштабируемость и повышают стоимость проектирования информационных систем. В связи с этим автоматизация процесса нормализации реляционных баз данных является актуальной научно-технической задачей, решение которой позволит повысить качество и эффективность проектирования баз данных.

Цель работы

Цель работы — разработать метод нормализации в реляционных базах данных с использованием анализа функциональных зависимостей.

Задачи

Для достижения поставленной цели выделены следующие задачи:

- Провести анализ предметной области реляционных баз данных;
- Рассмотреть подходы к автоматической нормализации в реляционных базах данных;

— Формализовать постановку задачи в виде функциональной модели;
— Разработать метод автоматической нормализации в реляционных базах данных;
— Изложить особенности предлагаемого метода;
— Сформулировать и описать ключевые шаги метода в виде схем алгоритмов;
— Описать структуры данных, используемые в алгоритмах;
— Описать взаимодействие отдельных частей системы;
— Обосновать выбор программных средств реализации предложенного метода;
— Описать взаимодействие пользователя с программным обеспечением;
— Исследовать характеристики разработанного метода.

1. Аналитическая часть

1.1 Анализ предметной области

1.1.1 Основы реляционной модели данных

Реляционная модель данных, предложенная Эдгаром Франком Коддом в 1970 году, представляет собой логическую модель организации данных, основанную на математических принципах теории множеств и логики предикатов первого порядка. Основная идея реляционной модели заключается в представлении всех данных в базе данных в виде набора отношений (часто визуализируемых как таблицы), где каждая строка представляет собой запись, а каждый столбец — атрибут этой записи. Связи между данными устанавливаются не через физические указатели, а через значения в общих атрибутах [6].

Ключевым преимуществом реляционной модели является обеспечение высокого уровня независимости данных, как логической, так и физической [5, 7]. Логическая независимость данных означает, что общая структура базы данных (концептуальная схема) может быть изменена без необходимости переписывать прикладные программы. Физическая независимость данных подразумевает, что способ хранения данных и методы доступа к ним могут быть изменены без воздействия на концептуальную схему или приложения. Такая структура позволяет упростить проектирование баз данных, обеспечить целостность и непротиворечивость хранимой информации, а также предоставить мощные и гибкие средства для извлечения и манипулирования данными с использованием языков запросов, таких как SQL (Structured Query Language) [6, 8].

Ниже приведены ключевые понятия реляционной модели данных [9]:

— **Понятие домена**

  Домен (англ. Domain) — это множество допустимых значений, которые может принимать атрибут. Домены определяют типы данных (например, строка, целое число, дата) и обеспечивают ограничение значений при внесении данных в таблицу.

Обозначим множество доменов как:

$$D = \{D_1, D_2, \dots, D\}, \quad (1)$$

где каждый D_i — это непустое множество возможных значений для элементов соответствующего типа.

— **Атрибут**

Атрибут (англ. Attribute) — именованная переменная, характеризующая одну из характеристик предметной области. Каждый атрибут соотносится с одним из доменов.

Пусть A — множество имен атрибутов:

$$A = \{A_1, A_2, \dots, A_n\}. \quad (2)$$

Для каждого атрибута A_i определяется соответствующий домен $dom(A_i) \in D$. Таким образом, допустимое значение атрибута A_i должно принадлежать множеству $dom(A_i)$.

— **Кортеж**

Кортеж — упорядоченный набор значений атрибутов, представляющий одну запись (строку) в отношении. Если имеется n атрибутов $A_1, A_2, \dots, A_n$, то кортеж можно записать как

$$t = (v_1, v_2, \dots, v_n), \quad (3)$$

где $v_i \in dom(A_i)$ для каждого $i = 1, 2, \dots, n$.

Множество всех возможных кортежей, соответствующих атрибутам $A_1, A_2, \dots, A_n$, есть декартово произведение доменов:

$$dom(A_1) \times dom(A_2) \times \dots \times dom(A_n). \quad (4)$$

Каждый конкретный кортеж t относится к некоторому отношению и обозначается как $t[A_1], t[A_2], \dots, t[A_n]$ для указания значений по атрибутам.

— **Отношение**

Отношение — множество кортежей, обладающих одинаковой структурой атрибутов. Формально отношение R определяется как пара:

$$R = (U, r), \tag{5}$$

где

— $U = \{A_1, A_2, ..., A_n\}$ — конечное множество атрибутов, называемое схемой отношения;

— $r \subseteq dom(A_1) \times dom(A_2) \times ... \times dom(A_n)$ — конечное множество кортежей t, где $t = (v_1, v_2, ..., v_n)$ и $v_i \in dom(A_i)$.

Число атрибутов n называется арностью отношения. Иногда отношение обозначают просто $R(A_1, A_2, ..., A_n)$, подчёркивая его схему.

Помимо упомянутых понятий, реляционная модель опирается на реляционную алгебру — набор формальных операций над отношениями, позволяющих задавать запросы и преобразования. Операции в ней эквивалентны операциям из теории множеств, только применяются к множествам кортежей [9, 10].

1.1.2 Функциональные зависимости

Функциональные зависимости (сокр. ФЗ) являются ключевым формализмом для описания семантических связей между атрибутами в реляционных базах данных [3]. Они отражают правила, по которым значение одного или нескольких атрибутов однозначно определяет значение других атрибутов в одном и том же отношении. Анализ ФЗ позволяет выявить скрытую логику предметной области и служит основой для процедур нормализации, направленных на устранение избыточности и предотвращение аномалий обновления [3].

Понятие функциональной зависимости

Функциональная зависимость $X \to Y$ в отношении $R(U)$ означает, что для любых двух кортежей t_1 и t_2 из r выполнено:

$$\textit{если } t_1[X] = t_2[X], \textit{ то } t_1[Y] = t_2[Y], \quad (6)$$

где X *и* Y — непустые подмножества множества атрибутов U[11]. Проекция кортежа t на множество X обозначается как $t[X]$, а значение атрибута A в кортеже t — как $t[A]$. Говорят, что X определяет Y или что Y функционально зависит от X. Формально:

- Пусть $U = \{A_1, A_2, \dots, A_n\}$ — множество атрибутов отношения R;
- Пусть $X \subseteq U$ и $Y \subseteq U$;
- Говорят, что X функционально определяет Y (обозначают $X \to Y$) тогда и только тогда, когда:

$$\forall t_1, t_2 \in r\text{: если } \forall A \in X\text{: } t_1[A] = t_2[A], \text{то } \forall B \in Y\text{: } t_1[B] = t_2[B]. \quad (7)$$

В этом случае X называют детерминантом, а Y — зависимой частью.

Функциональная зависимость является формализованным выражением бизнес-правила предметной области [3]. Например, в отношении *Студенты(Идентификатор, Имя, ДатаРождения)* зависимость *{Идентификатор} → {Имя, ДатаРождения}* отражает правило «по идентификатору студента однозначно определяется его имя и дата рождения».

Аксиомы Армстронга

Для формального вывода новых функциональных зависимостей из заданного множества F используют аксиомы, предложенные Армстронгом, которые гарантируют полноту и непротиворечивость вывода [12]:

1. **Рефлексивность** (англ. Reflexivity):

Если $Y \subseteq X$, то $X \rightarrow Y$;

2. **Усиление** (англ. Augmentation):

Если $X \rightarrow Y$, то $XZ \rightarrow YZ$ для любого $Z \subseteq U$;

3. **Транзитивность** (англ. Transitivity):

Если $X \rightarrow Y$ и $Y \rightarrow Z$, то $X \rightarrow Z$.

Кроме этих базовых аксиом зачастую используются производные правила вывода:

— **Композиция** (англ. Composition):

Если $X \rightarrow Y$ и $X \rightarrow Z$, то $X \rightarrow YZ$;

— **Декомпозиция** (англ. Decomposition):

Если $X \rightarrow YZ$, то $X \rightarrow Y$ и $X \rightarrow Z$;

— **Псевдотранзитивность** (англ. Pseudotransitivity):

Если $X \rightarrow Y$ и $YZ \rightarrow W$, то $XZ \rightarrow W$.

Множество аксиом Армстронга обеспечивает полное и корректное логическое следование зависимостей: любые зависимости, выводимые из F при помощи аксиом, действительно выполняются в любом отношении, удовлетворяющем F, и все зависимости, выполняющиеся в любом таком отношении, могут быть выведены [12, 13].

Замыкание множества атрибутов

Для практического анализа функциональных зависимостей часто вычисляют замыкание множества атрибутов X относительно набора зависимостей F [11]. Замыкание X^+ — это множество атрибутов, функционально определяемых из X через F:

$$X^+ = \{ A \in U \mid X \rightarrow A \text{ выводимо из } F \}. \quad (8)$$

Алгоритм вычисления X^+:

1. Инициализировать $X^+ = X$.
2. Пока есть зависимость $Y \rightarrow Z$ в F такая, что $Y \subseteq X^+$ и $Z \notin X^+$: добавить к X^+ все атрибуты из Z.
3. Повторять, пока новое добавление возможно.

Результат X^+ содержит все атрибуты, функционально зависящие от X. Замыкание используется, чтобы проверить, является ли X суперключом (если $X^+ = U$) либо выявить, функционально ли зависит один атрибут от другого.

Суперключи и кандидатные ключи

Множество атрибутов X называется суперключом в отношении $R(U)$, если $X^+ = U$. Это означает, что значения всех атрибутов отношения определяются значениями атрибутов X [11, 14].

При этом кандидатный ключ — это минимальное по включению множество атрибутов $K \subseteq U$, являющееся суперключом, то есть $K^+ = U$, и для любого $Y \subset K$ выполняется $Y^+ \neq U$ [14].

Алгоритм поиска всех кандидатных ключей:

1. Выделить все атрибуты U.
2. Сгенерировать потенциальные комбинации атрибутов $X \subseteq U$.
3. Для каждой комбинации вычислить X^+. Если $X^+ = U$, то X является суперключом.
4. Отсеять те суперключи, которые не являются минимальными (то есть существуют собственные подмножества, являющиеся суперключами).

Данный алгоритм при больших n атрибутов имеет экспоненциальную сложность, однако на практике количество комбинаций ограничено контекстом задачи [15].

Так называемый «первичный ключ» выбирается из множества кандидатных ключей для конкретного отношения и используется СУБД в качестве основного средства идентификации кортежей.

Минимальный (канонический) базис функциональных зависимостей

Для эффективного хранения и анализа ФЗ часто строят минимальный (канонический) набор зависимостей — такое представление исходного множества F, где каждая зависимость представлена в простейшей форме без избыточных атрибутов [16].

Критерии минимального базиса:

1. Правая часть каждой зависимости состоит из одного атрибута.
2. В левой части не содержится лишних атрибутов (то есть удаление любого атрибута приводит к тому, что зависимость уже не будет выводиться из набора).
3. Из набора не удалена ни одна зависимость (не существует зависимости, выводимой из остальных).

Алгоритм построения минимального покрытия F^{m} из исходного множества F:

1. Разложение. Для каждой зависимости $X \to Y \in F$ с $|Y| > 1$ заменить её на набор зависимостей $X \to A$ для каждого $A \in Y$.
2. Удаление лишних атрибутов в левой части. Для каждой зависимости $X \to A$ рассмотреть каждый атрибут $B \in X$: вычислить $(X \setminus \{B\})^+$ относительно текущего F. Если $(X \setminus \{B\})^+$ содержит A, то удалить B из левой части зависимости.
3. Удаление избыточных зависимостей. Для каждой зависимости $f \in F^{\mathrm{m}}$ проверить, выводится ли f из множества $F^{\mathrm{m}} \setminus \{f\}$. Если да — удалить f.

В результате получается минимальный (канонический) базис F^{m}, однозначно определяющий исходное множество зависимостей [16].

Классы эквивалентности множеств зависимостей

Два множества зависимостей F и G называются эквивалентными, если для любого отношения $R(U)$: R удовлетворяет F тогда и только тогда, когда R удовлетворяет G [17]. Эквивалентность обеспечивается сохранением функциональных «следствий» набора зависимостей.

Чтобы проверить эквивалентность F и G, иногда используют сравнение их минимальных покрытий: минимальные покрытия F^{m} и G^{m} совпадают (с точностью до порядка зависимостей), если F и G эквивалентны.

Полная и транзитивная функциональная зависимость

Для анализа нормальных форм необходимо различать два типа зависимостей:

1. Полная функциональная зависимость. Зависимость $X \to A$ называется полной, если A не функционально определяется никаким собственным подмножеством X [18].

   Формально: $X \to A$ полная, если для любого $B \subset X$: $B \to A$ не выводится из F.

2. Транзитивная зависимость. Зависимость $X \to Z$ называется транзитивной, если существует некоторое атрибут Y, такое что $X \to Y$ и $Y \to Z$ при этом Y не является частью X и Y не является A [18]. Транзитивная зависимость приводит к тому, что значение A определяется через промежуточный атрибут, а не напрямую от ключа.

Понимание разницы между полной и транзитивной зависимостью будет необходимо в дальнейшем для определения второй и третьей нормальных форм.

1.1.3 Проблема избыточности и аномалии обновления данных

Одной из фундаментальных проблем, возникающих при неоптимальном проектировании реляционной базы данных, является избыточность данных. Избыточность не только приводит к нерациональному использованию дискового пространства, но и, создает предпосылки для возникновения так называемых аномалий обновления данных [3]. Эти аномалии представляют собой логические ошибки и несоответствия, которые могут проявиться при выполнении операций вставки, удаления или модификации данных, тем самым нарушая их согласованность и достоверность.

Избыточность данных

Избыточность данных в контексте реляционных баз данных означает дублирование одной и той же информации в нескольких местах, либо в рамках одного отношения (таблицы), либо в нескольких различных отношениях [19]. Хотя некоторая степень контролируемой избыточности может целенаправленно вводиться для повышения производительности запросов (например, в денормализованных схемах для хранилищ данных), неконтролируемая избыточность, является серьезным недостатком [19, 20].

Последствия избыточности проявляются в виде аномалий трех видов:

1. **Аномалия вставки** (англ. insertion anomaly). Данная аномалия возникает, когда для добавления нового объекта в базу данных необходимо вводить данные, которые фактически ещё не известны или могут являться избыточными [3]. На рисунке 1.1 приведен пример аномалии вставки для отношения *Экзамены(Студент, Дисциплина, Преподаватель, оценка)*: если обновить значение поля «*Преподаватель*» только в строке 3, возникнет рассогласованность — два преподавателя на один предмет.

Рисунок 1.1 – Пример аномалии вставки

2. **Аномалия удаления** (англ. deletion anomaly). Данная аномалия возникает при попытке удалить запись возникает риск утраты ценной информации [3]. К примеру, если в отношении *Экзамены(Студент, Дисциплина, Преподаватель, Оценка)* удалить все сдачи студентов по «*Базам данных*», потеряется связь с преподавателем по этой дисциплине. Пример продемонстрирован на рисунке 1.2.

	Студент	Дисциплина	Преподаватель	Оценка
1	Иванов И. И	Теория вероятностей	Земцова А. Ю.	4
2	Петров П. П	Базы данных	Марьина Е. А.	4
3	Иванов И. И	Базы данных	Марьина Е. А.	5

Удалить информацию о результатах по "Базам данных"

Рисунок 1.2 – Пример аномалии удаления

3. **Аномалия обновления** (англ. update anomaly). Данная аномалия возникает, когда изменение одного значения может потребовать одновременного модифицирования нескольких записей [3, 21]. К примеру, в отношении *Экзамены(Студент, Дисциплина, Преподаватель, Оценка)* нельзя добавить дисциплину без добавления сдающих ее студентов. Пример продемонстрирован на рисунке 1.3.

	Студент	Дисциплина	Преподаватель	Оценка
1	Иванов И. И	Базы данных	Марьина Е. А.	5
2	Иванов И. И	Теория вероятностей	Земцова А. Ю.	4
3	Петров П. П	Базы данных	Марьина Е. А.	4

Обновить на "Карпова З. А"

Рисунок 1.2 – Пример аномалии обновления

Все перечисленные аномалии существенно затрудняют поддержание базы данных в актуальном и согласованном состоянии, снижают достоверность хранимой информации и могут приводить к принятию неверных решений на основе этих данных [21]. Устранение избыточности и предотвращение аномалий обновления являются основными целями процесса нормализации реляционных баз данных [3].

1.1.4 Нормализация реляционных отношений

Нормализация реляционных отношений представляет собой процесс приведения структуры отношения к такому виду, при котором устраняются избыточность и аномалии обновления данных [3, 6]. Основу нормализации составляет концепция декомпозиции без потерь (англ. lossless decomposition), предполагающая разложение исходного отношения на несколько более простых отношений с сохранением возможности восстановить исходные данные через операции соединения без появления ложных или потерянных кортежей [3, 6, 17].

Декомпозиция без потерь как фундамент нормализации

Процесс нормализации начинается с анализа исходной схемы отношения R и множества функциональных зависимостей F, описывающих семантические связи между атрибутами. Декомпозиция отношения R на два или более подотношений $R_1, R_2, \dots, R_k$ называется декомпозицией без потерь, если после разложения и последующего объединения (через естественное соединение) результирующее множество кортежей полностью совпадает с исходным, без добавления новых или удаления существующих данных [3, 17].

Критерием корректности декомпозиции без потерь является наличие хотя бы одной зависимости $X \rightarrow Y$ из F, такой что X содержит атрибуты, общие для R_1 и R_2, и при этом X совместно с зависимостями гарантирует восстановление всех исходных кортежей. Именно декомпозиция без потерь обеспечивает целостность данных после нормализации: она запрещает утрату информации, гарантируя, что никакая пара кортежей не объединится искусственно и никакая исходная запись не будет утрачена при разложении [17].

Общая идея нормальных форм

Нормальные формы (сокр. НФ) — это формальные критерии, задающие уровни структурной «идеализации» отношения с точки зрения отсутствия избыточности. Каждая следующая нормальная форма вводит более жёсткие ограничения на функциональные зависимости и структуру отношений [3, 6, 17].

Цель применения нормальных форм к проектированию баз данных:

— Гарантировать, что в отношении отсутствуют аномалии вставки, удаления и обновления.
— Обеспечить минимальное дублирование данных.
— Сохранить семантическую корректность и целостность информации в системе.

Нормализация выполняется поэтапно: сначала схема отношения проверяется на соответствие первой нормальной форме, затем — второй, третьей и более высоким. Если на каком-то этапе отношение не удовлетворяет критериям НФ, оно декомпозируется на более мелкие подотношения, удовлетворяющие данным критериям [18, 22].

Определение первой нормальной формы (1НФ)

Первая нормальная форма накладывает требование **атомарности** всех значений атрибутов. Это означает, что каждый атрибут в отношении должен содержать единственное скалярное значение в каждой ячейке; недопустимо хранение списков, множественных значений или повторяющихся групп в одном атрибуте [18, 28]. Таким образом, условием для 1НФ является следующее правило: всё множество атрибутов U отношения R должно быть разделено на такие, что каждое значение $t[A_i]$ представляет собой неделимый одиночный элемент данных.

Если у отношения есть атрибуты, допускающие неоднородные или составные значения, то для приведения к первой нормальной форме выполняется декомпозиция: составные атрибуты разбиваются на отдельные столбцы, а повторяющиеся группы представляются в виде дополнительных строк [18, 28].

Определение второй нормальной формы (2НФ)

Вторая нормальная форма накладывает дополнительное ограничение на отношение, находящееся в 1НФ. Она касается частичных функциональных зависимостей от составного ключа: отношение R, удовлетворяющее 1НФ, приведено ко 2НФ, если каждый неключевой атрибут полностью функционально зависит от всего (каждого) составного ключа, а не от его части [3, 17].

Требования 2НФ:

— Сначала необходимо определить все возможные кандидатные ключи отношения R.

— Если существует зависимость $X \rightarrow A$, где X — часть ключа (то есть X не равно полному ключу, но $X \subset K$, где K — кандидатный ключ), и A — неключевой атрибут, то отношение не удовлетворяет второй нормальной форме.

Для устранения частичной зависимости выполняют декомпозицию: исходное отношение разбивается таким образом, чтобы в одном подотношении оставался составной ключ вместе с атрибутами, полностью зависящими от него, а частично зависящие переносятся в новое отношение, где они будут полностью зависеть от собственной части ключа [3, 18].

Определение третьей нормальной формы (3НФ)

Третья нормальная форма расширяет требования 2НФ, исключая транзитивные зависимости. Отношение R, находящееся во второй нормальной форме, приведено в третью нормальную форму, если каждый неключевой атрибут функционально зависит только от каждого кандидатного ключа напрямую, а не опосредованно через другой неключевой атрибут [3, 18].

Требования 3НФ:

— Отношение должно быть в 2НФ.

— Нет зависимостей вида $K \rightarrow B$ и $B \rightarrow C$, где K — кандидатный ключ, B — некоторый неключевой атрибут, а C — другой неключевой атрибут.

Для устранения транзитивных зависимостей также применяют декомпозицию без потерь: атрибуты, зависящие транзитивно, переносятся в отдельные подотношения вместе с промежуточными атрибутами, так чтобы сохранить прямые зависимости от ключей.

Определение нормальной формы Бойса-Кодда (НФБК)

Нормальная форма Бойса–Кодда является более строгой версией 3НФ. Отношение R удовлетворяет BCNF, если для любой функциональной зависимости $X \rightarrow Y$, действующей в отношении, множество X является суперключом (то есть X содержит некоторый кандидатный ключ целиком) [3, 18]. Требование BCNF: если в отношении R действует зависимость $X \rightarrow Y$, то $X \supseteq K$, где K — кандидатный ключ отношения R.

Таким образом, каждая зависимость информации должна исходить от надёжно идентифицирующего набора атрибутов. Если же найдена зависимость, где левый набор X не является суперключом, значит, отношение не в BCNF. Устранение нарушения BCNF выполняется декомпозицией: исходную схему разбивают на подотношения, в которых все зависимости будут исходить от ключа.

Определение четвертой нормальной формы (4НФ)

Четвёртая нормальная форма вводит понятие многозначных зависимостей (сокр. МФЗ) и направлена на устранение случаев, когда один атрибут независимо множественно зависит от ключа [3, 18].

МФЗ $X \twoheadrightarrow Y$ в отношении $R(U)$ означает, что для любого набора значений X и каждого значения из Y, набор значений остальных атрибутов $Z = U \setminus (X \cup Y)$ может принимать все комбинации независимо от Y [18, 28]. Проще говоря, значения атрибута Y не зависят от значений остальных атрибутов, кроме X. Это приводит к тому, что в таблице начинают образовываться независимые сочетания значений Y и Z для одного и того же X, что составляет избыточность.

Отношение R приведено к 4НФ, если оно находится в BCNF и не содержит ненулевых МФЗ [3, 18].

В случае нарушения 4НФ выполняют декомпозицию отношения по МФЗ: отношение делится на такие подотношения, чтобы каждый независимый набор атрибутов мог существовать отдельно, без создания повторяющихся комбинаций.

1.2 Обзор существующих методов автоматической нормализации

Для обзора существующих методов рассмотрим три инструмента, реализующих метод автоматической нормализации реляционных баз данных: Micro, RDBNorma, JMathNorm и подход на основе генетических алгоритмов. Каждый из методов реализует автоматизированное обнаружение функциональных зависимостей и разложение схемы отношений на нормальные формы, но отличается выбором алгоритмических приёмов, структурой представления данных и степенью взаимодействия с пользователем.

1.2.1 Micro

Micro — это программный инструмент с оконным интерфейсом, предназначенный для автоматической нормализации схемы реляционной базы данных. Интерфейс Micro обеспечивает визуализацию матрицы зависимостей и соответствующего ориентированного графа, что позволяет пользователю понять структуру связей между атрибутами ещё до генерации нормализованных таблиц [22].

В основе Micro лежит алгоритм построения матрицы зависимостей, после чего автоматически выполняется разложение на 2НФ, 3НФ и НФБК [22, 23]. Сначала система вычисляет замыкание каждого множества атрибутов в матрице, затем на основе полученных зависимостей строится ориентированный граф, в котором узлы соответствуют атрибутам, а дуги отражают наличие функциональной зависимости. Затем алгоритм последовательно выделяет микрокомпоненты схемы, удовлетворяющие требованиям нормальных форм, и генерирует итоговые отношения, параметризованные выбранной нормальной формой. Данный инструмент также позволяет экспортировать результаты в SQL-код для дальнейшего использования в системах управления базами данных. Тем не менее, при работе с крупными схемами Micro демонстрирует ограничения по потребляемым ресурсам [22, 23].

1.2.2 RDBNorma

RDBNorma — полуавтоматизированный инструмент, направленный на оптимизацию памяти и времени при нормализации на уровне схемы. В основе RDBNorma лежит представление схемы отношения и набора функциональных зависимостей в едином односвязном списке, что позволяет значительно уменьшить затраты оперативной памяти по сравнению с Micro [24]. Такой подход упрощает хранение всех атрибутов и зависимостей, так как односвязный список позволяет хранить информацию о ключе, неключевых атрибутах и соответствующих зависимостях в компактной форме [24].

Алгоритмы RDBNorma реализовывают преобразование отношения в 2НФ и 3НФ без потерь информации посредством итеративного анализа списка зависимостей. Сначала система разбивает каждую зависимость на отдельные пары «лево → право», затем удаляет избыточные атрибуты из левой части, вычисляя замыкание для каждой потенциальной зависимости. Если атрибут в левой части оказывается лишним, он удаляется. После минимизации набора зависимостей производится проверка на полноту и декомпозиция, которая реализует разложение с сохранением свойств безошибочной соединимости и сохранения зависимостей [24].

При сравнении RDBNorma с Micro отмечено, что RDBNorma работает в 2,89 раза быстрее и требует примерно вдвое меньше памяти для представления одной и той же схемы и набора зависимостей. Это достигается за счёт односвязного списка, который хранит атрибуты и зависимости в компактном формате, вместо матрицы зависимостей или деревьев [24].

1.2.3 JMathNorm

JMathNorm — интерактивный инструмент, разработанный для применения возможностей Mathematica при нормализации реляционных схем [25]. Основная идея JMathNorm заключается в использовании ядерного языка Mathematica для выполнения фундаментальных операций над множествами, таких как вычисление замыкания множества атрибутов, получение минимального покрытия зависимостей, а также проверка атрибута на принадлежность к кандидату в ключи. Таким образом, реализуется грамматика для реляционных операций.

В JMathNorm реализованы модули для приведения схемы в 2НФ, 3НФ и НФБК. Пользователю достаточно задать список атрибутов и функциональных зависимостей через GUI, после чего система автоматически берёт зависимости, преобразует их в минимальное покрытие и выполняет разложение без потерь. Благодаря тому, что Mathematica оптимизирована для работы с булевой алгеброй и операциями на множествах, вычислительные этапы выполняются эффективно, а результаты отображаются в удобочитаемом виде, включая графики, таблицы и формулы [25]. При этом JMathNorm поддерживает расширение функциональности за счёт дополнительных модулей Mathematica, что упрощает добавление новых нормальных форм или специализированных проверок схем [25].

1.2.4 Генетический алгоритм

Подход на основе генетических алгоритмов представляет собой нетрадиционную методику нормализации, в которой схема базы данных выводится не из заранее заданного набора зависимостей, а выявляется посредством оптимизации данных [26]. Таким образом, генетический алгоритм работает с исходными бизнес-данными и стремится найти такое представление схемы отношений, при котором достигается минимальная избыточность при сохранении семантической состоятельности [26]. Каждый геном в популяции кодирует возможную схему, включая разбиение исходного набора атрибутов на подотношения и варианты функциональных зависимостей.

Инициализация популяции производится случайным образом: создаются несколько возможных схем-декомпозиций, из которых каждая оценивается по функции приспособленности [26]. В качестве функции потерь выступает комбинированный критерий, включающий степень сохранения информационной целостности (декомпозиция без потерь, сохранение зависимостей) и минимизацию избыточности, измеряемую как количество повторяющихся значений атрибутов в отношении к полной схеме. В результате эволюционного процесса алгоритм обычно сходится к схеме, близкой к оптимальной по критериям нормализации. В проведенных исследованиях алгоритм восстановил около 72% исходных схем без предварительных знаний о зависимостях [26, 27]. Данный подход особенно актуален для задач, где отсутствуют формализованные зависимости, или где их нельзя формализовать на практике в принципе.

1.3 Сравнение существующих методов автоматической нормализации

Критерии сравнения

Исходя из анализа существующих методов, были сформулированы следующие критерии для сравнения методов автоматической нормализации (Micro, RDBNorma, JMathNorm, Генетический алгоритм):

1. Максимально поддерживаемая нормальная форма. Данный критерий служит показателем, до какой нормальной формы алгоритм может привести отношение автоматически.
2. Наличие графического интерфейса (GUI). Данный критерий показывает, есть ли у данного инструмента графический интерфейс.
3. Необходимость явного задания списка функциональных зависимостей. Данный критерий служит показателем, может ли метод самостоятельно выявлять функциональные зависимости непосредственно из данных.

Результаты сравнения методов

Результаты сравнения существующих методов автоматической нормализации в реляционных базах данных представлены в таблице 1.

Таблица 1. – Результаты сравнения методов.

Метод	**Макс. нормальная форма**	**Графический интерфейс**	**Явное задание ФЗ**
Micro	НФБК	+	+
RDBNorma	3НФ	+	+
JMathNorm	НФБК	+	+
Генетический алгоритм	3НФ	-	-

Выводы из сравнения методов

В результате проведенного сравнения можно сделать вывод, что рассмотренные алгоритмы не реализуют автоматическую нормализацию выше уровня НФБК.

1.4 Формальная постановка задачи

Исходя из анализа предметной области и рассмотренных существующих методов автоматической нормализации, задача автоматической нормализации в реляционных базах данных с использованием анализа функциональных зависимостей была формализована в виде нотации IDEF0 нулевого уровня. Формальная постановка задачи изображена на рисунке 1.

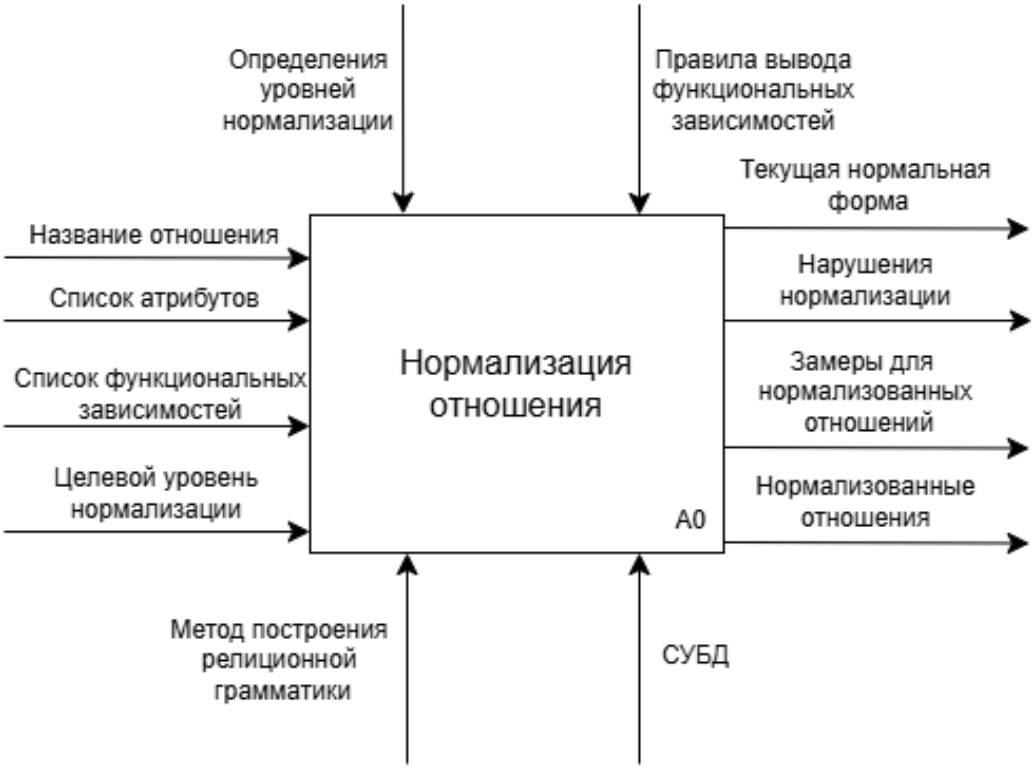

Рисунок 1. – Формальная постановка задачи

Выводы по аналитической части

В рамках аналитической части, был проведен анализ предметной области реляционных баз данных, проведен обзор существующих решений нормализации реляционных баз данных, проведено их сравнение, а также формализована постановка задачи автоматической нормализации в виде функциональной модели.

2. Конструкторская часть

2.1 Ключевые шаги метода

Спроектированный метод автоматической нормализации состоит из трех ключевых этапов:

1. Анализ исходного отношения. На этом этапе для введенного отношения атрибуты разделяются на простые и непростые, находятся кандидатные ключи, минимальное покрытие, замыкания, текущий уровень нормализации и нарушения нормализации для следующего уровня.
2. Нормализация отношения. На данном этапе, исходя из данных, полученных в ходе выполнения анализа исходного отношения, строятся декомпозированные к целевой нормальной форме отношения через устранения нарушений условий нормальных форм вплоть до целевой.
3. Замер характеристик нормализованных отношений. На данном этапе, проводятся замеры используемой памяти в зависимости от уровня нормализации, и тестирование на декомпозицию без потерь.

Ключевые шаги разработанного метода представлены на диаграмме IDEF0 первого уровня, изображенной на рисунке 2.1.

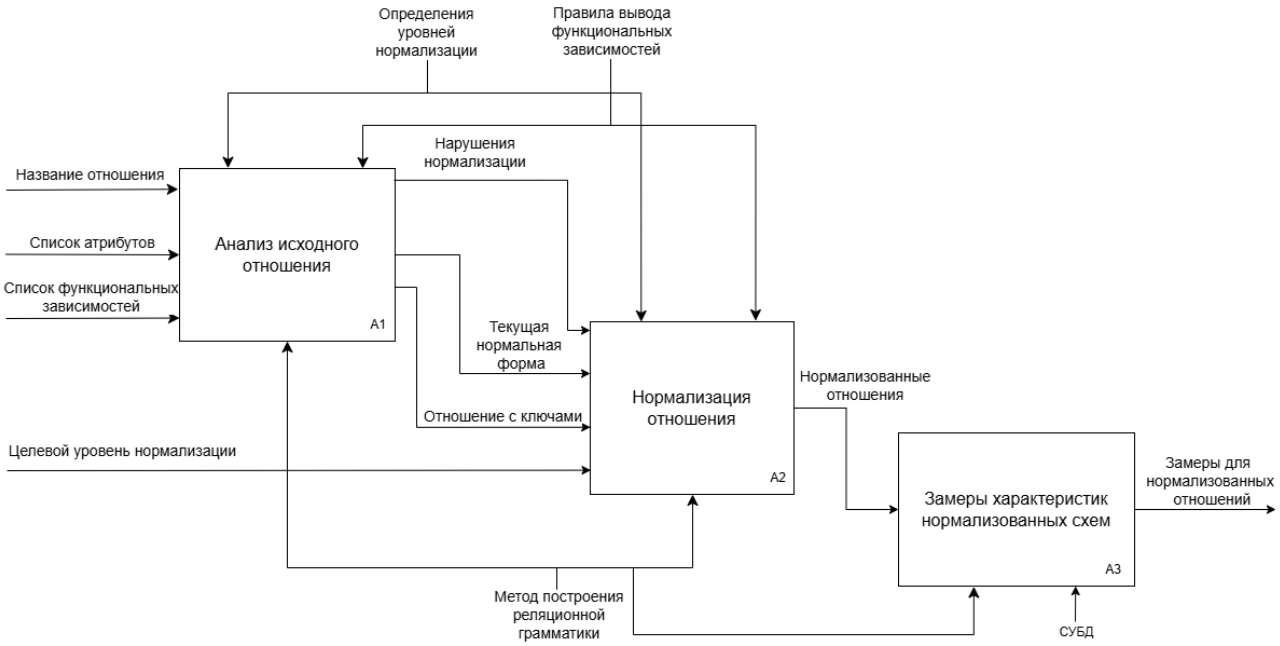

Рисунок 2.1 – Диаграмма IDEF0 первого уровня

Шаг анализа исходного отношения состоит из трех ключевых алгоритмов.

1. Алгоритм поиска замыкания для подмножеств множества атрибутов отношения.
2. Алгоритм поиска кандидатных ключей отношения.
3. Общий алгоритм анализа исходного отношения.

Алгоритм поиска замыкания для подмножеств множества атрибутов отношения представлен на рисунке 2.2.

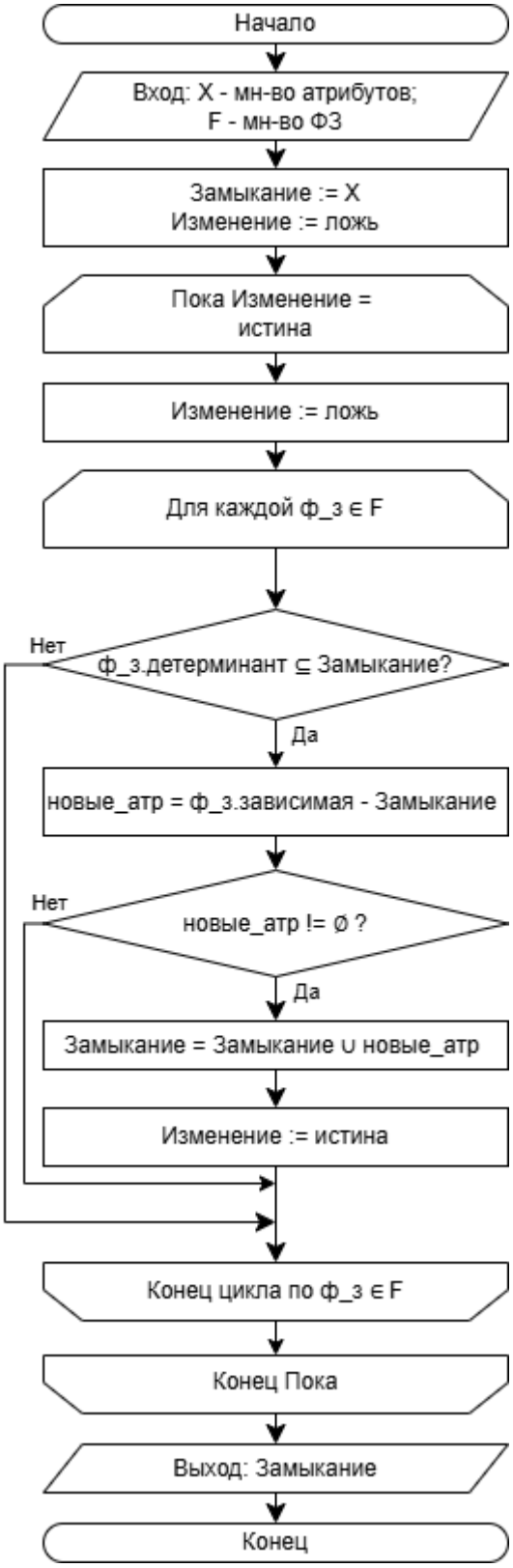

Рисунок 2.2 – Алгоритм поиска замыкания

Алгоритм поиска кандидатных ключей отношения представлен на рисунках 2.3 и 2.4.

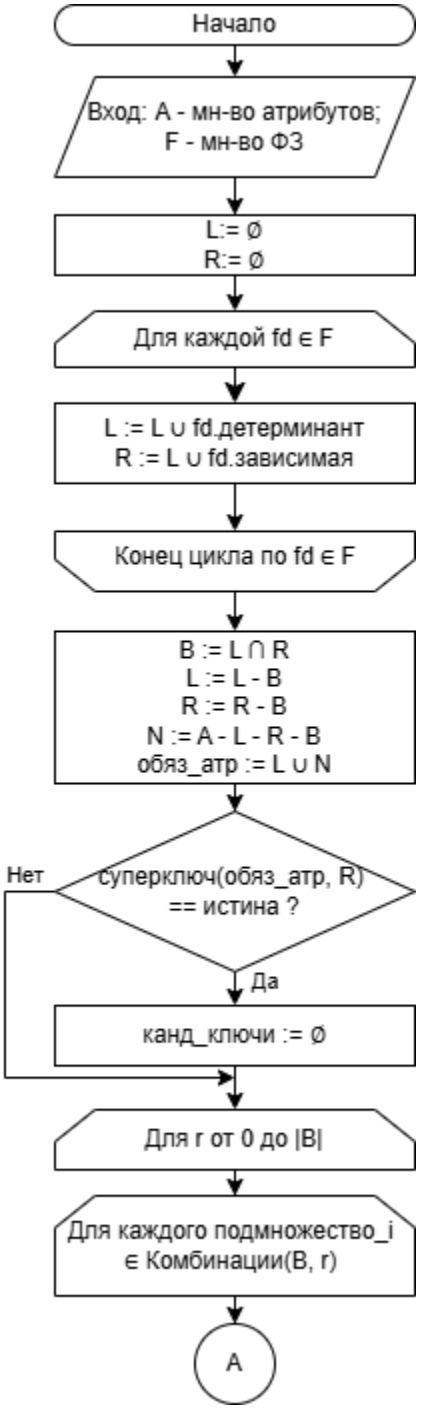

Рисунок 2.3 – Алгоритм поиска кандидатных ключей, часть 1

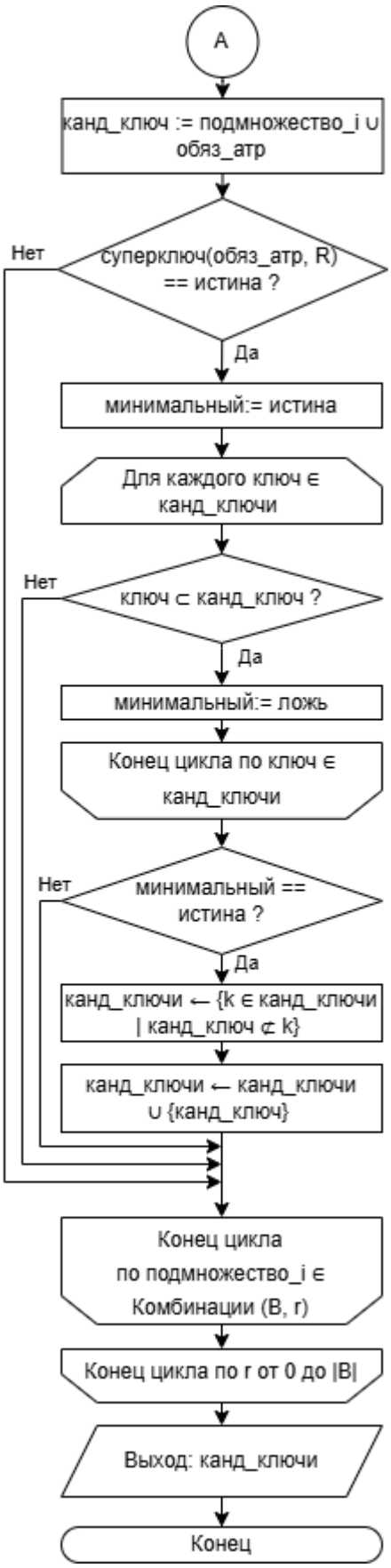

Рисунок 2.4 – Алгоритм поиска кандидатных ключей, часть 2

Алгоритм анализа исходного отношения представлен на рисунке 2.5.

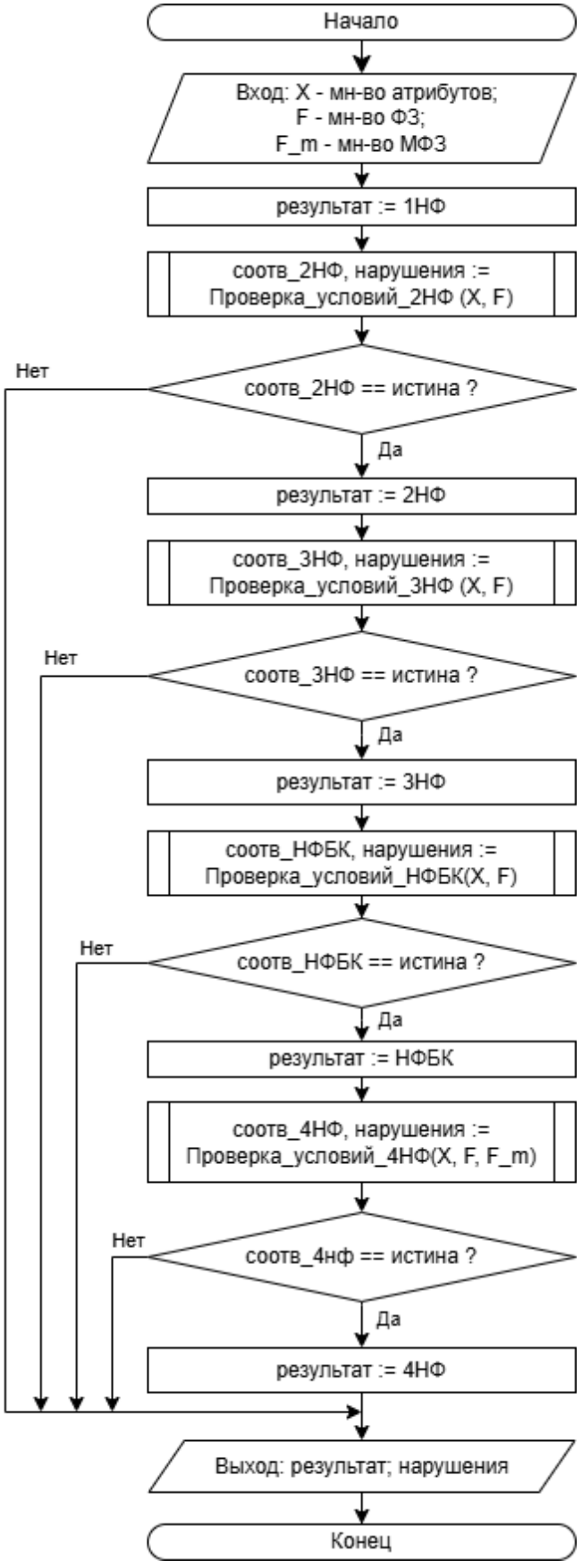

Рисунок 2.5 – Алгоритм анализа исходного отношения, часть 2

Шаг декомпозиции отношения состоит из алгоритма декомпозиции отношения, в котором устраняются нарушения целевой нормальной формы, полученные на шаге анализа исходного отношения. Алгоритм декомпозиции отношения представлен на рисунке 2.6.

Рисунок 2.6 – Алгоритм декомпозиции отношения

2.2 Взаимодействие отдельных частей системы

Взаимодействие отдельных частей системы отражено на схеме данных, представленной на рисунке 2.2.

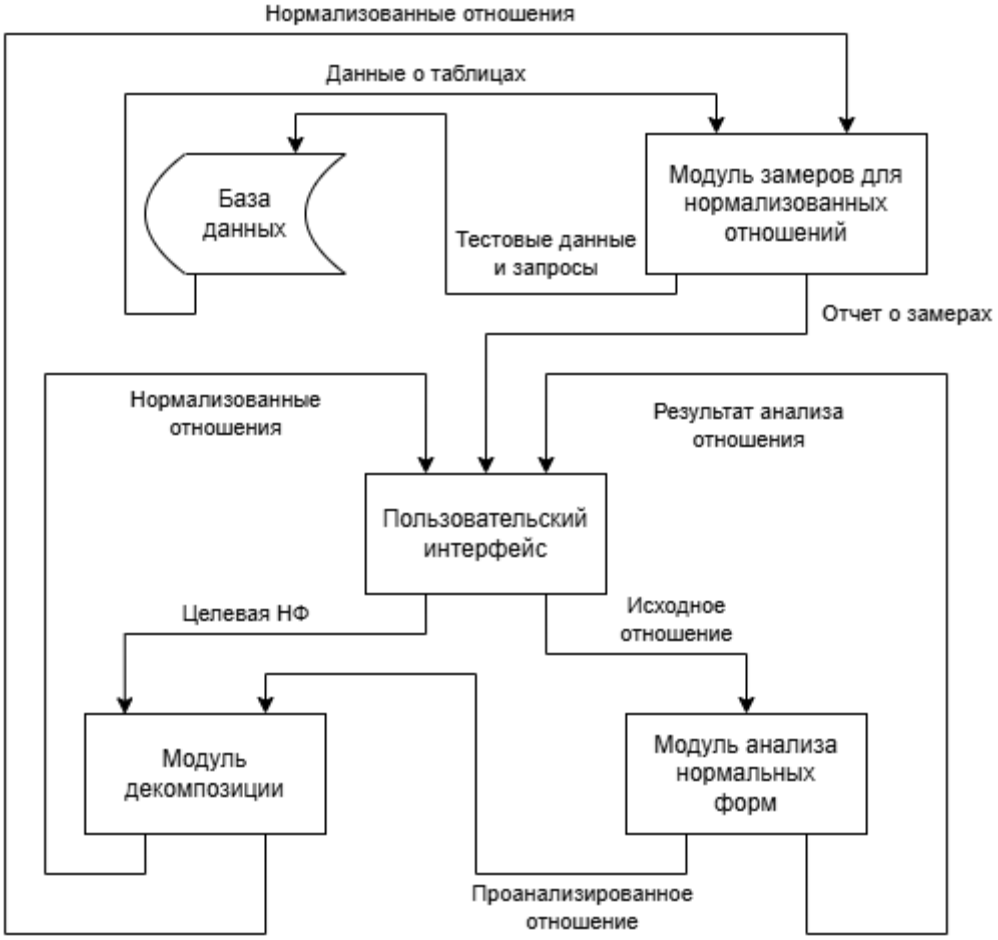

Рисунок 2.2 – Взаимодействие отдельных частей системы

2.3 Алгоритм тестирования метода

На рисунке 2.3 изображен алгоритм функционального тестирования разработанной реализации метода автоматической нормализации в реляционных базах данных.

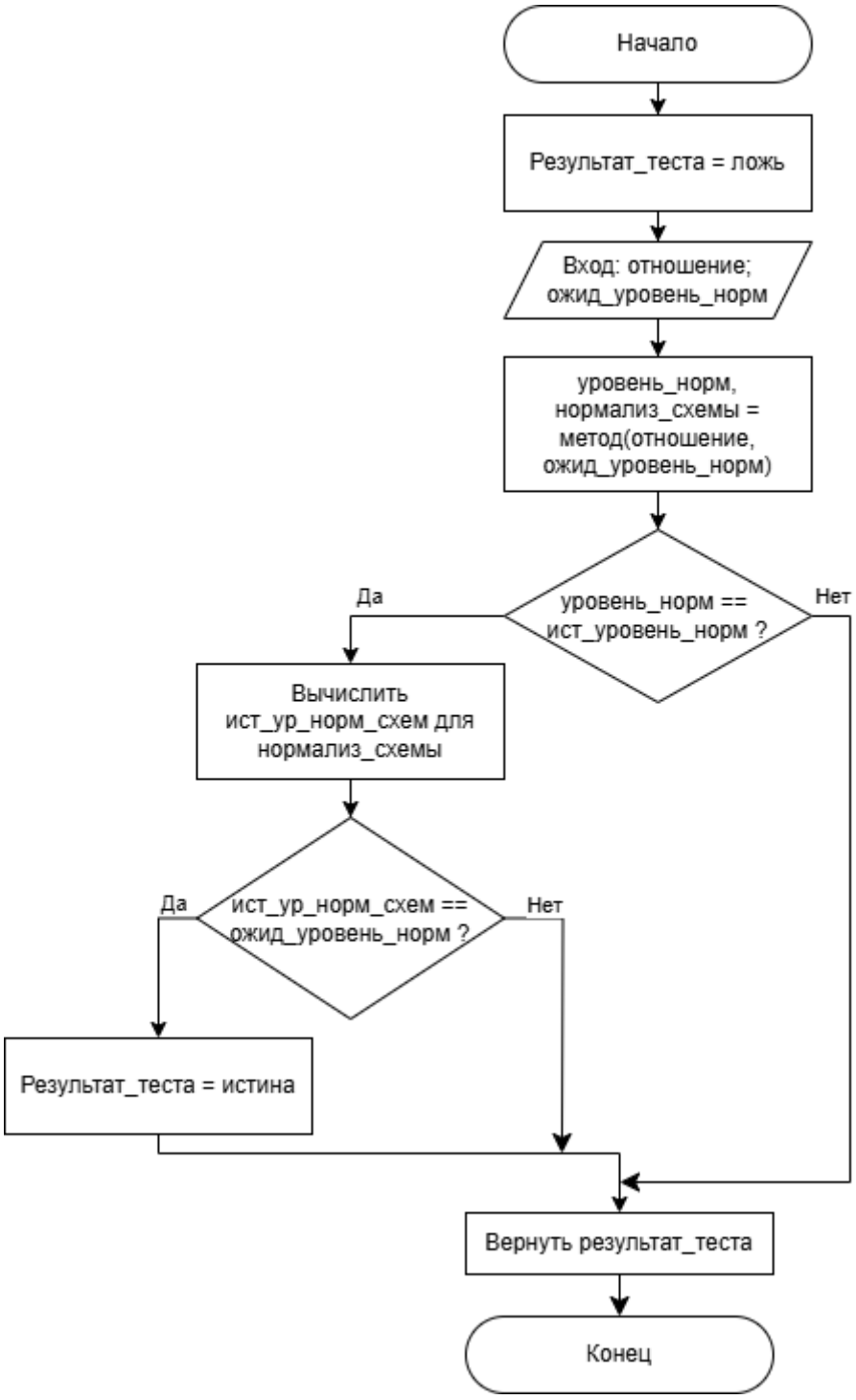

Рисунок 2.3 – Схема алгоритма функционального тестирования

2.4 Классы эквивалентности тестирования метода

В результате анализа предметной области для тестирования метода автоматической нормализации были выделены следующие группы классов эквивалентности:

1. Группа классов эквивалентности по исходной нормальной форме отношения представлена в таблице 2.1.

Таблица 2.1. – Группа классов эквивалентности по исходной нормальной форме отношения.

Описание класса эквивалентности	**Ожидаемый результат работы метода**
Отношение в целевой НФ	НФ отношения определена корректно, декомпозиции отношения к целевой НФ не происходит
Исходное отношение в 1НФ, есть нарушение 2НФ	НФ отношения определена как 1НФ, в нормализованных отношениях устранены частичные зависимости
Исходное отношение в 2НФ, есть нарушение 3НФ	НФ отношения определена как 2НФ, в нормализованных отношениях устранены транзитивные зависимости
Исходное отношение в 3НФ, есть нарушение НФБК	НФ отношения определена как 3НФ, в нормализованных отношениях устранены ФЗ, где детерминант не является суперключом.
Исходное отношение в НФБК, есть нарушение 4НФ	НФ отношения определена как 4НФ, в нормализованных отношениях устранены многозначные ФЗ

2. Группа классов эквивалентности по структуре и количеству ключей представлена в таблице 2.2.

Таблица 2.2. – Группа классов эквивалентности по структуре и количеству ключей.

Описание класса эквивалентности	**Ожидаемый результат работы метода**
В отношении один простой кандидатный ключ	Кандидатный ключ находится корректно
В отношении один составной кандидатный ключ	Кандидатный ключ находится корректно
Несколько пересекающихся кандидатных ключей	Кандидатные ключи находятся корректно
Несколько непересекающихся кандидатных ключей	Кандидатные ключи находятся корректно

3. Группа классов эквивалентности по потере ФЗ при декомпозиции представлена в таблице 2.3.

Таблица 2.3. – Группа классов эквивалентности по потере ФЗ при декомпозиции

Описание класса эквивалентности	**Ожидаемый результат работы метода**
Декомпозиция без потери зависимостей	При нормализации все исходные ФЗ сохранены в результирующих нормализованных отношениях
Декомпозиция с потерей зависимостей	При нормализации потеряны ФЗ, нарушающие целевую НФ

4. Группа классов эквивалентности по количеству и типу атрибутов и ФЗ представлена в таблице 2.4.

Таблица 2.4. – Группа классов эквивалентности по количеству и типу атрибутов и ФЗ.

Описание класса эквивалентности	**Ожидаемый результат работы метода**
Отношение без атрибутов	Программа сообщает, что для анализа требуются атрибуты
Отношение с атрибутами, но без заданных ФЗ	НФ отношения определяется как 4НФ, декомпозиция не требуется
Отношение с одним атрибутом	НФ отношения определяется как 4НФ, декомпозиция не требуется
Отношение, где заданы только тривиальные ФЗ	НФ отношения определяется как 4НФ, декомпозиция не требуется

Выводы по конструкторской части

В рамках конструкторской части был спроектирован метод автоматической нормализации, описаны его ключевые шаги в виде схем алгоритмов, описано взаимодействие отдельных частей системы, описан алгоритм функционального тестирования спроектированного метода, а также описаны классы эквивалентности для функционального тестирования метода.

3. Технологическая часть

3.1 Выбор средств программной реализации

В качестве языка программирования для реализации метода автоматической нормализации был выбран Python. Выбор аргументирован тем, что для данного языка существуют библиотеки, такие как Typing, реализующие необходимые для алгоритмов реляционной алгебры типы данных и операции над ними [28, 29]. К тому же, Python поддерживает объектно-ориентированную парадигму программирования, которая удобна для реализации связей между компонентами реляционной алгебры.

Дополнительно для взаимодействия с СУБД PostgreSQL был выбран драйвер psycopg2, обеспечивающий надёжное и высокопроизводительное исполнение SQL-запросов и управление транзакциями на уровне приложения [30]. Графический интерфейс разработан с использованием стандартного модуля tkinter, что позволяет получить кроссплатформенное решение без привлечения внешних GUI-фреймворков и минимизировать количество зависимостей [31]. Для визуализации результатов анализа и тестирования выполнена интеграция с библиотекой matplotlib, обладающей широким спектром возможностей построения научных графиков и активной поддержкой в сообществе [32].

3.2 Взаимодействие пользователя с интерфейсом

Для начала работы, пользователю необходимо ввести схему отношения: имя атрибутов, их тип (или оставить по умолчанию), первичные ключи. Далее, пользователь вводит функциональные зависимости по введенным атрибутам. После этого становятся доступны модули анализа, декомпозиции и замеров.

Графический интерфейс для ввода входных данных отображен на рисунке 3.1.

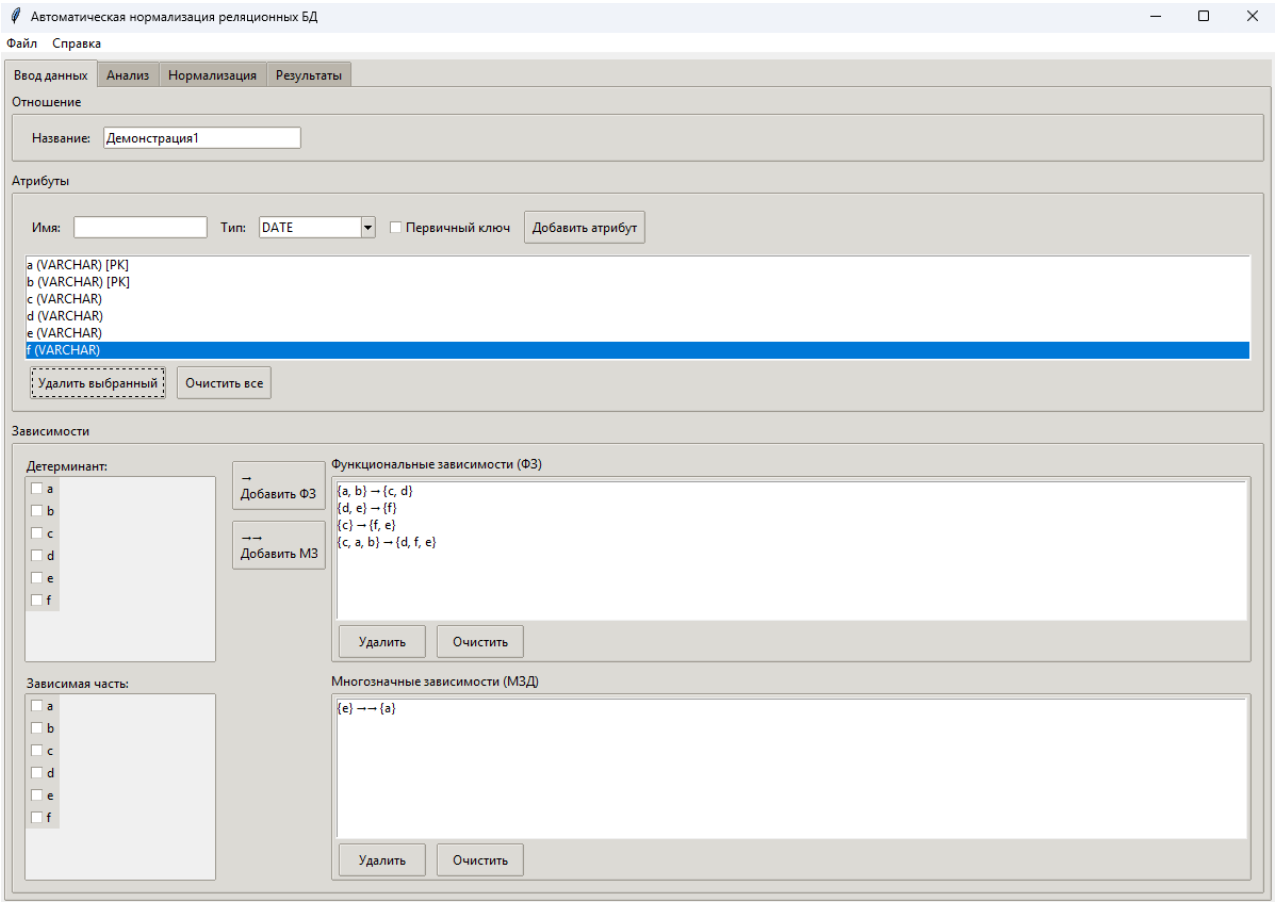

Рисунок 3.1 – Графический интерфейс для ввода входных данных

Графический интерфейс для модуля анализа исходного отношения представлен на рисунке 3.2.

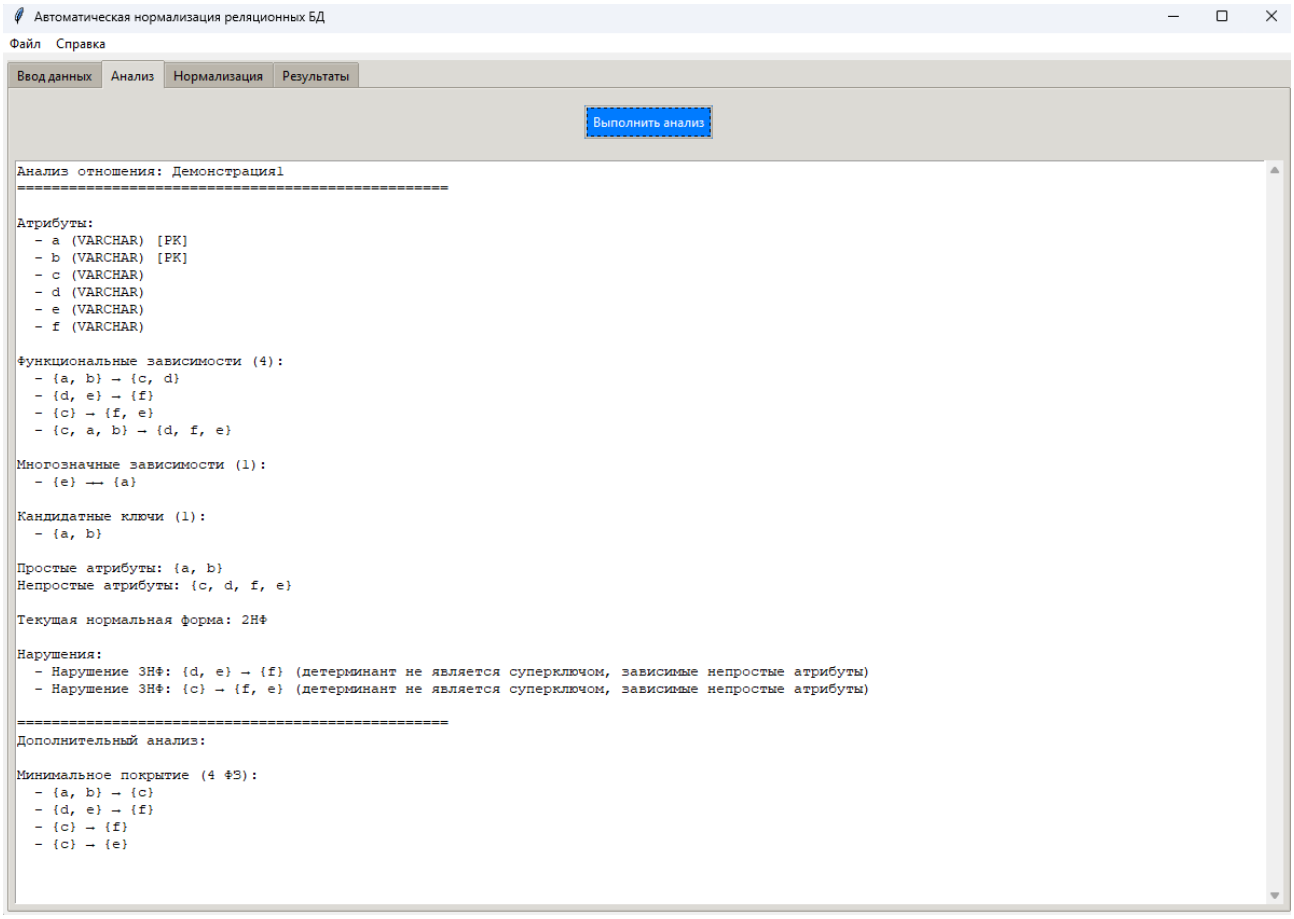

Рисунок 3.2 – Графический интерфейс для модуля анализа исходного отношения

Графический интерфейс для модуля нормализации отношений представлен на рисунке 3.3.

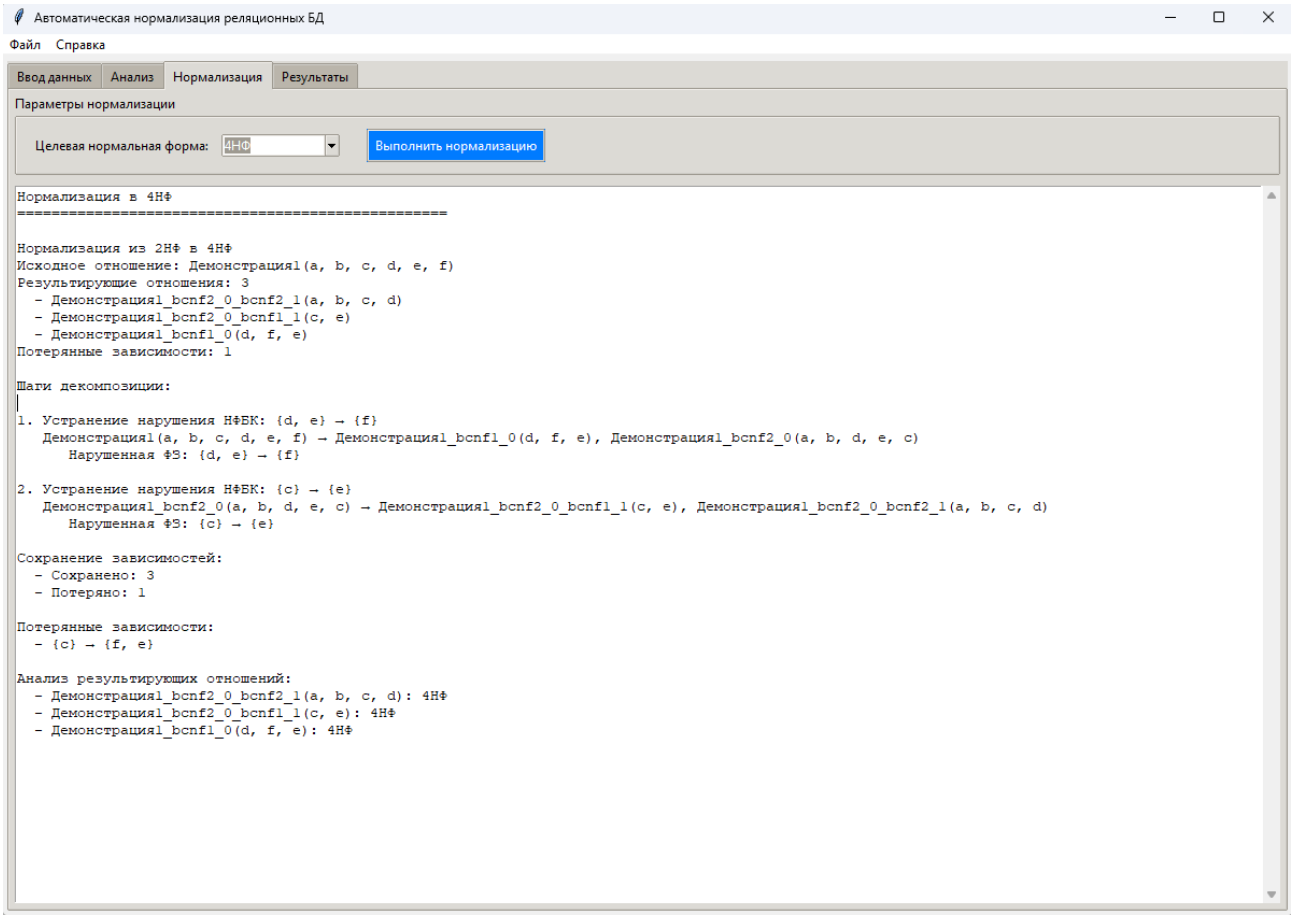

Рисунок 3.3 – Графический интерфейс для модуля нормализации отношений

Графический интерфейс для модуля замеров характеристик нормализованных отношений представлен на рисунке 3.4.

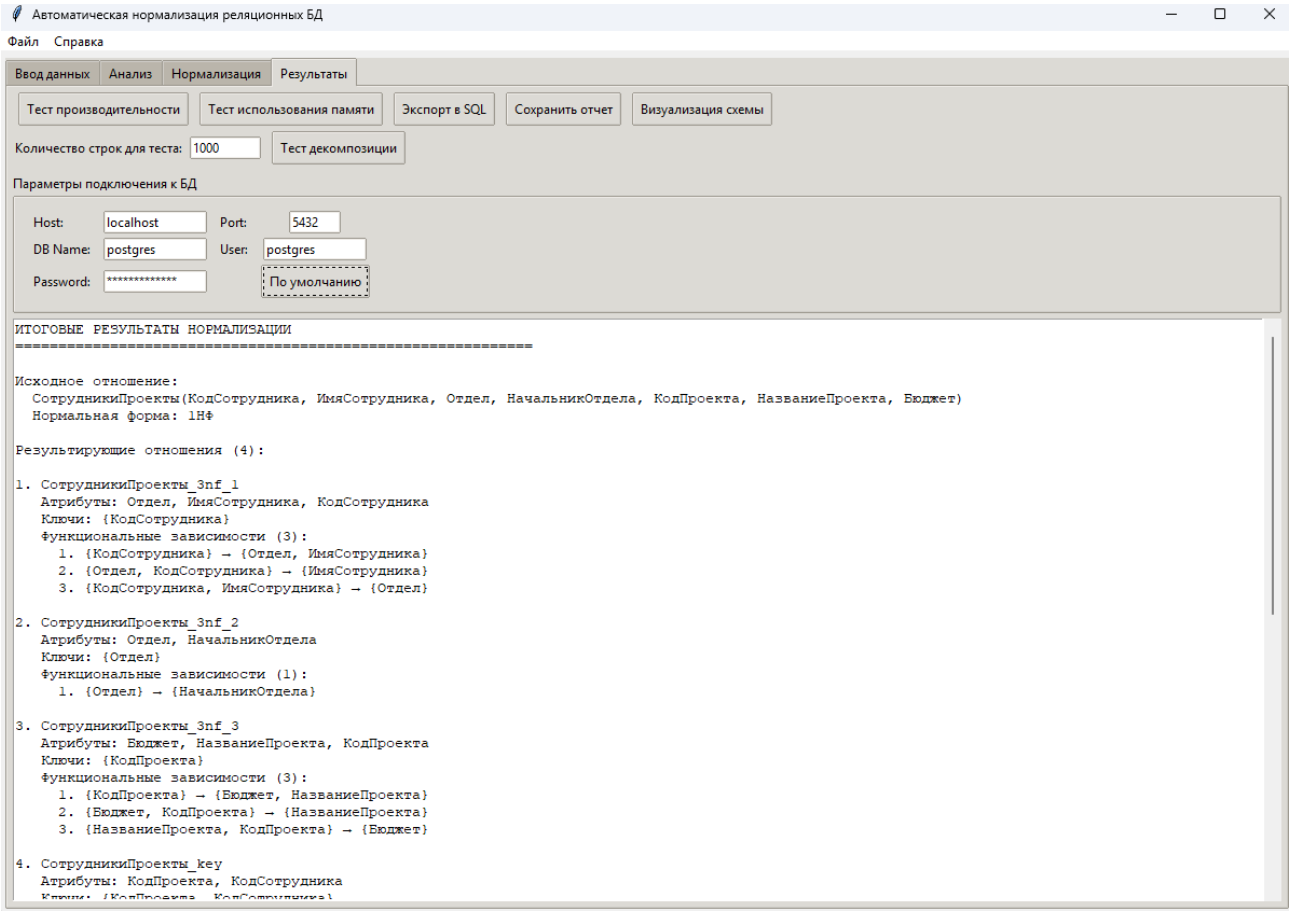

Рисунок 3.3 – Графический интерфейс для модуля замеров характеристик нормализованных отношений

3.3 Результаты тестирования программного обеспечения

Ниже приведены результаты тестирования разработанной реализации метода автоматической нормализации, объединенные по группам классов эквивалентности.

1. Результаты тестирования по группе классов эквивалентности по исходной нормальной форме отношения представлен в таблице 3.1.

Таблица 3.1. – Результат тестирования по группе классов эквивалентности по исходной нормальной форме отношения.

Класс эквивалентности и входные данные	**Ожидаемый результат**	**Результат работы метода**	**Соответствие результата ожидаемому**
Отношение уже в целевой НФ (НФБК). Отношение: *R(a(PK), b, c)*. ФЗ: $a \rightarrow (b, c)$. Целевой уровень: НФБК	НФ: НФБК. R1: *(a(PK), b, c)*.	НФ: НФБК. *R1*: *R(a(PK), b, c)*.	+
Нарушение 2НФ. Отношение: *R(a(PK), b(PK), c, d, e)*. ФЗ: $(a, b) \rightarrow (e)$; $(b) \rightarrow (c, d)$. Целевой уровень: 2НФ	НФ: 1НФ Нарушение: $(b) \rightarrow (c, d)$. R1: *(b, c, d)*. R2: *(a, b, e)*.	НФ: 1НФ Нарушение: $(b) \rightarrow (c, d)$. R1: *(b, c, d)*. R2: *(a, b, e)*.	+
Нарушение 3НФ. Отношение: *R(a(PK), b, c)*. ФЗ: $a \rightarrow b$; $b \rightarrow c$. Целевой уровень: 3НФ	НФ: 3НФ. Нарушение: $a \rightarrow b$; $b \rightarrow c$ R1: *R(a(PK), b)* R2: *R(b(PK), c)*	НФ: 3НФ. Нарушение: $a \rightarrow b$; $b \rightarrow c$ R1: *R(a(PK), b)* R2: *R(b(PK), c)*	+

Продолжение таблицы 3.1. – Результат тестирования по группе классов эквивалентности по исходной нормальной форме отношения.

Нарушение НФБК. Отношение: *R(a(PK), b, c(PK), d(PK)).* ФЗ: $a \rightarrow b; b \rightarrow a$ Целевой уровень: НФБК	НФ: НФБК. *R1: R(a(PK), b)* *R2: R(a(PK), c, d)*	НФ: НФБК. *R1: R(a(PK), b)* *R2: R(a(PK), c, d)*	+
Нарушение 4НФ. Отношение: *R(a(PK), b, c).* ФЗ: $a \rightarrow b; a \rightarrow c.$ Многозначные зависимости: $b \rightarrow\rightarrow c.$ Целевой уровень: 4НФ	НФ: 4НФ. *R1: R(b(PK), c(PK))* *R2: R(a(PK), b)*	НФ: 4НФ. *R1: R(b(PK), c(PK))* *R2: R(a(PK), b)*	+

2. Результаты тестирования по группе классов эквивалентности по структуре и количеству ключей представлен в таблице 3.2.

Таблица 3.2. – Результат тестирования по группе классов эквивалентности по структуре и количеству ключей.

Класс эквивалентности и входные данные	**Ожидаемый результат**	**Результат работы метода**	**Соответствие результата ожидаемому**
Один простой кандидатный ключ. Отношение: *R(a, b, c).* ФЗ: $a \rightarrow (b, c).$	Кандидатные ключи: $\{a\}$	Кандидатные ключи: $\{a\}$	+
Один составной кандидатный ключ. Отношение: *R(a, b, c).* ФЗ: $(a, b) \rightarrow c.$	Кандидатные ключи: $\{a, b\}$	Кандидатные ключи: $\{a, b\}$	+

Продолжение таблицы 3.2. – Результат тестирования по группе классов эквивалентности по структуре и количеству ключей.

Несколько пересекающихся кандидатных ключей. Отношение: *R(a, b, c).* ФЗ: *(a, b) → c; (a, c) → b*	Кандидатные ключи: *{a, b}, {a, c}*	Кандидатные ключи: *{a, b}, {a, c}*	+
Несколько непересекающихся кандидатных ключей. Отношение: *R(a, b, c, d).* ФЗ: *(a, b) → (c, d); (c, d) → (a, b)*	Кандидатные ключи: *{a, b}, {c, d}*	Кандидатные ключи: *{a, b}, {c, d}*	+

3. Результаты тестирования по группе классов эквивалентности по потере ФЗ при декомпозиции представлен в таблице 3.3.

Таблица 3.3. – Результат тестирования по группе классов эквивалентности по потере ФЗ при декомпозиции.

Класс эквивалентности и входные данные	**Ожидаемый результат**	**Результат работы метода**	**Соответствие результата ожидаемому**
Декомпозиция без потери зависимостей. Отношение: *R(a(PK), b, c).* ФЗ: *a → b; a → c*	НФ: 3НФ. R1: *R(a(PK), b)* R2: *R(a(PK), c)* Сохранено зависимостей: 2 Потеряно зависимостей: 0	НФ: 3НФ. R1: *R(a(PK), b)* R2: *R(a(PK), c)* Сохранено зависимостей: 2 Потеряно зависимостей: 0	+

Продолжение таблицы 3.3. – Результат тестирования по группе классов эквивалентности по потере ФЗ при декомпозиции.

Декомпозиция с потерей зависимостей. Отношение: *R(a, b, c).* ФЗ: *(a, b) → c; c → a.*	НФ: НФБК. *R1: R1(c(PK), a)* *R2: R2(c(PK), b)* Сохранено зависимостей: 1 *(c → a)* Потеряно зависимостей: 1 *(a, b → c)*	НФ: НФБК. *R1: R1(c(PK), a)* *R2: R2(c(PK), b)* Сохранено зависимостей: 1 *(c → a)* Потеряно зависимостей: 1 *(a, b → c)*	+

4. Результаты тестирования по группе классов эквивалентности по количеству и типу атрибутов и ФЗ представлен в таблице 3.4.

Таблица 3.4. – Результат тестирования по группе классов эквивалентности по количеству и типу атрибутов и ФЗ.

Класс эквивалентности и входные данные	**Ожидаемый результат**	**Результат работы метода**	**Соответствие результата ожидаемому**
Отношение без атрибутов. Отношение: *R()*. ФЗ: —	Ошибка: для анализа требуются атрибуты.	Ошибка: для анализа требуются атрибуты.	+

Продолжение таблицы 3.4. – Результат тестирования по группе классов эквивалентности по количеству и типу атрибутов и ФЗ

Отношение с атрибутами, но без заданных ФЗ. Отношение: *R(a, b, c).* ФЗ: — Целевой уровень: 4НФ	НФ: 4НФ. *R1: R(a(PK), b(PK), c(PK))*	НФ: 4НФ. *R1: R(a(PK), b(PK), c(PK))*	+
Отношение с одним атрибутом. Отношение: *R(a).* ФЗ: — Целевой уровень: 4НФ	НФ: 4НФ. *R1: R(a(PK))*	НФ: 4НФ. *R1: R(a(PK))*	+
Отношение, где заданы только тривиальные ФЗ. Отношение: *R(a, b).* ФЗ: $a \rightarrow a;\ b \rightarrow b$	НФ: 4НФ. R1: *R(a(PK), b(PK))*	НФ: 4НФ. R1: *R(a(PK), b(PK))*	+

Выводы по конструкторской части

В рамках технологической части был разработан спроектированный метод автоматической нормализации для реляционных баз данных, был обоснован выбор средств программной реализации, описано взаимодействие пользователя с программным обеспечением, а также проведено функциональное тестирование разработанного метода по ранее описанным классам эквивалентности.

4. Исследовательская часть

4.1 Постановка и условия исследования

В данном разделе будет проведено исследование зависимости временных характеристик разработанной реализации метода автоматической нормализации реляционных баз данных от сложности входных данных, а также будет измерено время выполнения различных запросов к базе данных в зависимости от уровня нормализации.

В рамках исследования зависимости временных характеристик разработанной реализации метода автоматической нормализации реляционных баз данных от сложности входных данных будет замерено время выполнения анализа отношения и нормализации отношения до заданной целевой нормальной формы от следующих факторов:

1. Количество атрибутов в исходном отношении (N). В этом случае количество функциональных зависимостей фиксируется (M = 5), а N варьируется в рамках выбранного диапазона;
2. Количество функциональных зависимостей (ФЗ) в исходном отношении (M). В этом случае количество атрибутов фиксируется (N = 5), а M варьируется в рамках выбранного диапазона.

Замеры времени анализа и нормализации отношений в зависимости от сложности входных данных будут проведены 100 раз, а после будет взято среднее арифметическое время по каждому уровню сложности для построения графиков.

В рамках исследования зависимости занимаемой памяти таблицами базы данных от уровня нормализации выполняется следующий эксперимент. Сначала с помощью разработанных алгоритмов декомпозиции формируются пять схем: исходная (англ. original) и нормализованные вплоть до 2НФ, 3НФ, НФБК и 4НФ. Каждая схема представлена одним или несколькими отношениями, которые заполняются идентичным набором данных — исходная таблица создаётся и

наполняется фиксированным числом строк, а нормализованные отношения формируются посредством операторов «*INSERT … SELECT DISTINCT …*» из исходного отношения, гарантируя одинаковый объём исходной информации. После этого, средствами СУБД происходят замеры следующих метрик:

— Размер базы данных, равный сумме размеров всех хранимых отношений;
— Плотность данных, равная отношение общего размера к числу строк (байт на строку);
— Коэффициент сжатия, равный отношению исходного объёма к объёму после нормализации.

Для исследования занимаемой памяти было выбрано отношение *Сотрудники_компании(КодСотрудника, ИмяСотрудника, Отдел, НачальникОтдела, КодПроекта, НазваниеПроекта, БюджетПроекта)*, в котором выполняются следующие ФЗ:

— *{КодСотрудника}→{ИмяСотрудника,Отдел}*;
— *{Отдел}→{НачальникОтдела}*;
— *{КодПроекта}→{НазваниеПроекта, БюджетПроекта}*.

Для моделирования двух уровней избыточности исходного отношения используются следующие конфигурации:

1. Высокая избыточность:

— Уникальных отделов: 5
— Уникальных проектов: 10
— Уникальных начальников: 5
— Уникальных имён сотрудников: 200

2. Низкая избыточность:

— Уникальных отделов: 30
— Уникальных проектов: 28
— Уникальных начальников: 30
— Уникальных начальников: 30

Технические характеристики среды исследования

Технические характеристики компьютера, на котором было проведено исследование, представлены далее:

процессор AMD (R) Ryzen (TM) 5-7500f CPU 3.7 ГГц, 6 физических ядер и 12 логических [33];

— оперативная память 32.0 Гбайт;

— операционная система Windows 11 Pro 23H2 [34].

Во время исследования компьютер был включен в сеть электропитания и был нагружен только системными приложениями

4.2 Результаты исследования

Результаты исследования зависимости временных характеристик разработанной реализации метода автоматической нормализации реляционных баз данных от количества атрибутов отношения представлен на рисунке 4.1/

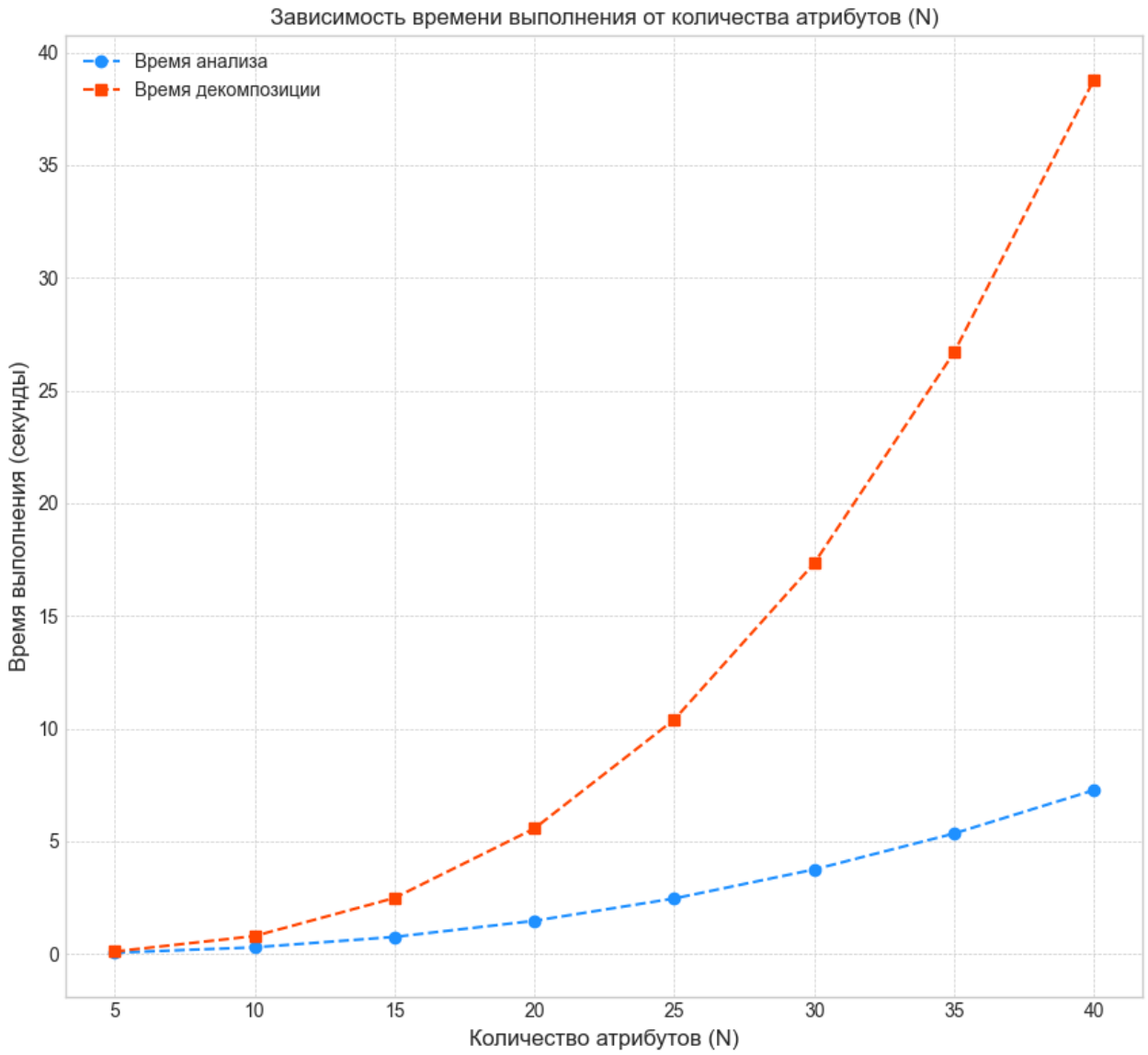

Рисунок 4.1. – График зависимости временных характеристик разработанной реализации метода от количества атрибутов.

Результат исследования зависимости временных характеристик разработанной реализации метода автоматической нормализации реляционных баз данных от количества функциональных зависимостей представлен на рисунке 4.2.

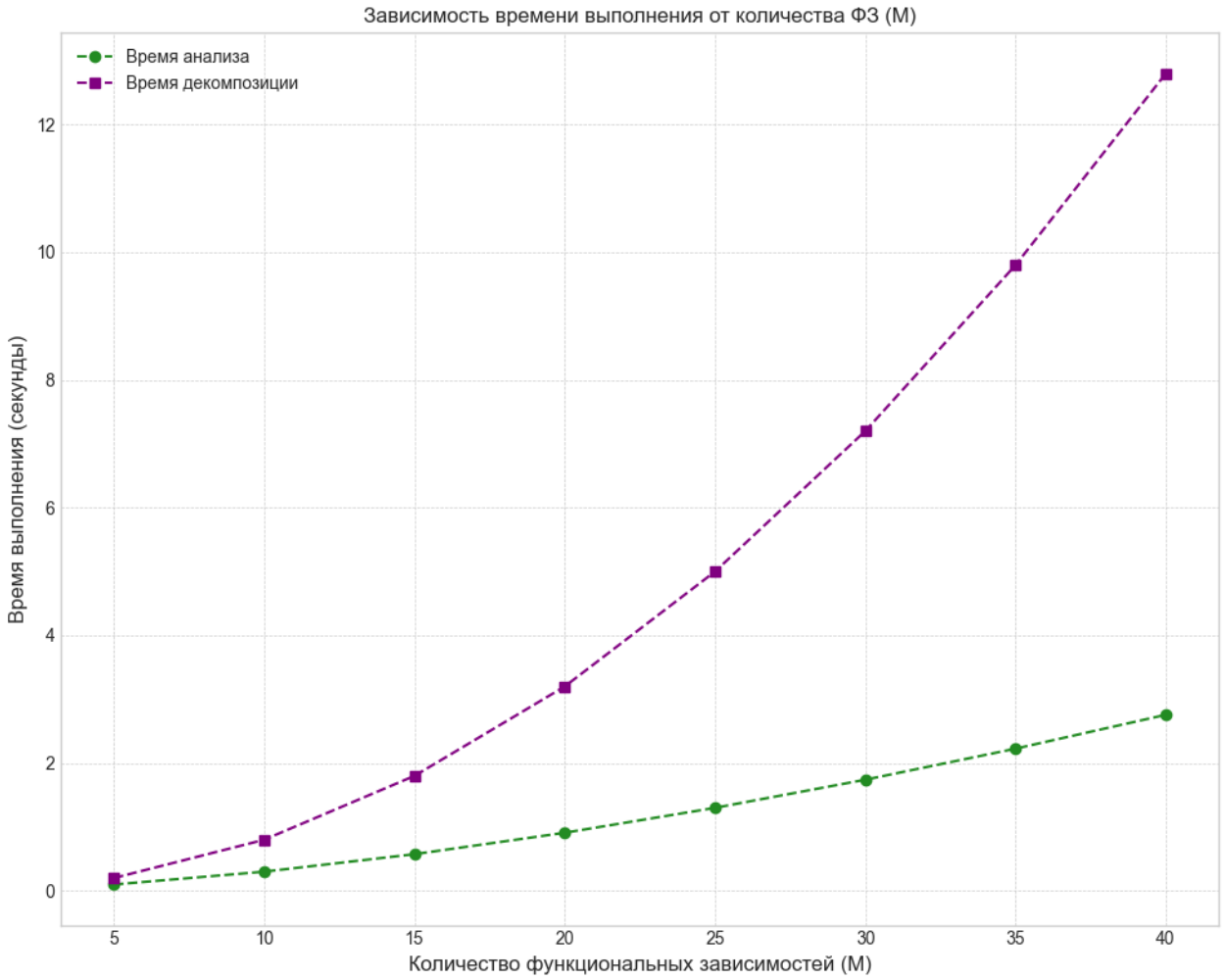

Рисунок 4.2. – График зависимости временных характеристик разработанной реализации метода от количества функциональных зависимостей.

Результат исследования зависимости занимаемой памяти таблицами базы данных от уровня нормализации для высокого уровня избыточности данных представлен на рисунке 4.3

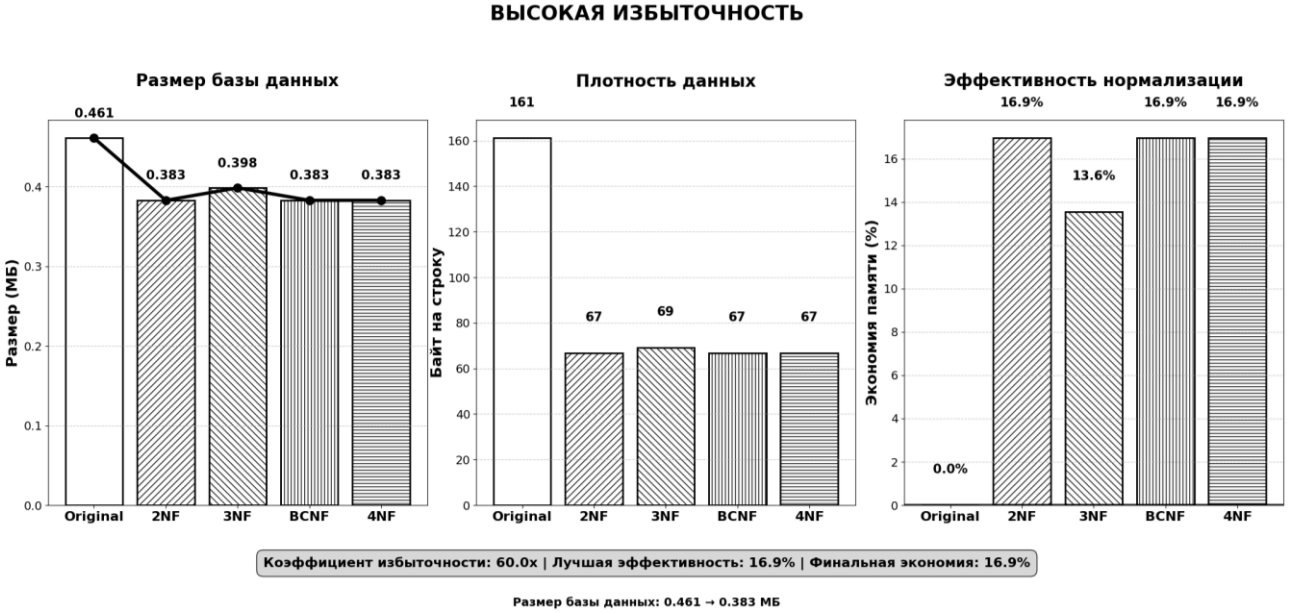

Рисунок 4.3. – Гистограмма зависимости занимаемой памяти таблицами базы данных от уровня нормализации для высокого уровня избыточности данных.

Результат исследования зависимости занимаемой памяти таблицами базы данных от уровня нормализации для высокого уровня избыточности данных представлен на рисунке 4.4

Рисунок 4.4. – Гистограмма зависимости занимаемой памяти таблицами базы данных от уровня нормализации для низкого уровня избыточности данных.

Выводы по исследовательской части

В рамках исследования зависимости временных характеристик разработанной реализации метода автоматической нормализации реляционных баз данных от сложности входных данных можно сделать вывод, что время анализа и нормализации зависит полиномиально и от количества атрибутов, и от количества функциональных зависимостей, так как алгоритмы анализа и декомпозиции имеют комбинаторный характер. Однако, исходя из полученных значений, можно утверждать, что показатель степени в зависимости временных характеристик метода от количества атрибутов больше, чем от количества функциональных зависимостей.

В рамках исследования времени выполнения различных запросов к базе данных в зависимости от уровня нормализации можно сделать вывод, что для выбранного отношения нормализация позволяет уменьшить объем хранимых данных. При этом эффективность нормализации в случае заполненности исходной таблицы с высокой избыточностью выше, чем при заполненности с низкой избыточностью.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была достигнута цель разработки метода автоматической нормализации в реляционных базах данных, а также были успешно выполнены все поставленные задачи:

— Проанализирована предметная область реляционных баз данных;

— Спроектирован метод автоматической нормализации в реляционных базах данных;

— Разработан спроектированный метод;

— Исследована зависимость времени анализа и декомпозиции отношений от количества атрибутов и функциональных зависимостей.

Направления дальнейшего развития для разработанного метода могут включать:

— Реализацию обратной композиции отношений;

— Вычисление характеристик нормализованных отношений для различных типов запросов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дмитриев А. П., Лейба С. Ш. Стремительный рост цифровых данных: анализ мировых трендов и прогноз развития в России // Региональная и отраслевая экономика. – 2024. – № 1. – С. 145–147.
2. Голиков О. И., Панкратов И. А. Исследование способов повышения эффективности обработки данных в реляционных БД на примере СУБД MySQL // Вестник Волжского университета им. В. Н. Татищева. — 2016. — Т. 2, № 2. — С. 124–130.
3. Элмасри Р., Наватха С. Б. Основы систем баз данных / пер. с англ. — М.: Вильямс, 2012. — 1008 с.
4. Донгаре Й. В., Дхабе П. С., Дешмукх С. В. RDBNorma: A semi-automated tool for relational database schema normalization up to third normal form // International Journal of Database Management Systems. — 2011. — Vol. 3, № 1. — С. 135–144.
5. Кодд Э. Ф. A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. — 1970. — Т. 13, №. 6. — С. 377–387.
6. Кодд Э. Ф. The relational model for database management: version 2 / Э. Ф. Кодд. — Reading (Mass.): Addison-Wesley, 1990. — 538 с.
7. Майер Д., Розенштейн Д., Салветер Ш. К., Стайн Дж., Уоррен Д. С. Toward logical data independence: A relational query language without relations // Proceedings of the ACM SIGMOD International Conference on Management of Data. — 1982. — New York: ACM Press, 1982. — P. 51–60.
8. Эльмасри Р. Logical and Physical Data Independence // Encyclopedia of Database Systems / под ред. Л. Лю, М. Т. Озсу. — New York: Springer, 2018. — С. 2136–2137.
9. Абитебул С., Халл Р., Виану В. Foundations of Databases: The logical level / S. Abiteboul, R. Hull, V. Vianu. — Reading (Mass.): Addison-Wesley Publishing Co., 1995. — 413 с.

10. Ульман Д. Д. Principles of Database and Knowledge-Base Systems. Volume I. Classical Database Systems / Д. Д. Ульман. — New York: Computer Science Press, 1988. — 672 с.
11. Сильберсчатц А., Корс Х. Ф., Сударшан С. Database System Concepts / A. Silberschatz, H. F. Korth, S. Sudarshan. — 6-е изд. — New York: McGraw-Hill, 2010. — С. 130–150.
12. Армстронг У. В. Dependency structures of data base relationships / W. W. Armstrong. — Stockholm: North-Holland, 1974. — С. 580–583.
13. Варди М. Ю. Fundamentals of dependency theory // Trends in Theoretical Computer Science / под ред. Э. Боргера. — Rockville, MD: Computer Science Press, 1987. — С. 171–224.
14. Рамакришнан Р., Герхке Й. Database Management Systems / R. Ramakrishnan, J. Gehrke. — 3-е изд. — Boston: McGraw-Hill, 2003. — С. 109–112.
15. Саидейан Х., Спенсер Т. An Efficient Algorithm to Compute the Candidate Keys of a Relational Database Schema // The Computer Journal. — 1996. — Т. 39, № 2. — С. 125–131.
16. Майер Д. Minimum Covers in the Relational Database Model // Journal of the ACM. — 1980. — Vol. 27, № 4. — С. 664–674.
17. Дейт К. Д. An Introduction to Database Systems / C. J. Date. — 8-е изд. — Boston: Addison-Wesley, 2003. — С. 142–144.
18. Кент У. A Simple Guide to Five Normal Forms in Relational Database Theory // Communications of the ACM. — 1983. — Т. 26, № 2. — С. 120–125.
19. Вигхио М. С., Ханзада Т. Ж., Кумар М. Analysis of the effects of redundancy on the performance of relational database systems // Proceedings of the 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS). — IEEE, 2017. — С. 1–6.
20. Шин С. К., Сандерс Г. Л. Denormalization strategies for data retrieval from data warehouses // Decision Support Systems. — 2006. — Т. 42, №. 1. — С. 267–282.

21. Пратиюшсах П. Database Normalization // International Journal for Research Publication & Seminar. — Т. 11, изд. 2, 2020. — С. 9–10.
22. Ду Х., Вэри Л. Micro: A normalization tool for relational database designers // Journal of Network and Computer Applications. — 1999. — Т. 22, № 4. — С. 215–232.
23. Бахмани А. Х., Нагхибзадех М., Бахмани Б. Automatic database normalization and primary key generation / A. H. Bahmani, M. Naghibzadeh, B. Bahmani. — In: Proceedings of the Canadian Conference on Electrical and Computer Engineering. — 2008. — IEEE, 2008. — С. 11–16.
24. Дхонгаре Ю. В., Дхабе П. С., Дешмук С. В. RDBNorma: A semi-automated tool for relational database schema normalization up to third normal form / Y. V. Dongare, P. S. Dhabe, S. V. Deshmukh. — International Journal of Database Management Systems. — 2011. — Т. 3, № 1. — С. 133–154.
25. Язици А., Каракая З. JMathNorm: A Database Normalization Tool Using Mathematica / A. Yazici, Z. Karakaya // Computational Science – ICCS 2007. – Berlin, Heidelberg: Springer, 2007. — Lecture Notes in Computer Science. Т. 4488. — С. 186–193.
26. Акадал Э., Сатман М. Х. A Novel Automatic Relational Database Normalization Method / Emre Akadal, Mehmet Hakan Satman. — Acta Informatica Pragensia. — 2022. — Т. 11, № 3. — С. 293–308.
27. Ван Боммел П., Люкасиус К. Б., Ван Дер Вийде Т. П. Genetic algorithms for optimal logical database design / P. van Bommel, C. B. Lucasius, Th. P. van der Weide. — Information and Software Technology. — 1994. — Т. 36, № 12. — С. 759–771.
28. Python Software Foundation. Python.org [Электронный ресурс]. – Режим доступа: https://www.python.org/ (дата обращения: 27.05.2025).
29. Python Software Foundation. typing — Support for type hints [Электронный ресурс]. — Режим доступа: https://docs.python.org/3/library/typing.html (дата обращения: 27.05.2025).

30. The PostgreSQL Global Development Group. PostgreSQL: The world’s most advanced opensource relational database [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/ (дата обращения: 27.05.2025).
31. Python Software Foundation. tkinter — Python interface to Tcl/Tk [Электронный ресурс]. — Режим доступа: https://docs.python.org/3/library/tkinter.html (дата обращения: 27.05.2025).
32. Matplotlib Developers. Matplotlib — Visualization with Python [Электронный ресурс]. – Режим доступа: https://matplotlib.org/ (дата обращения: 27.05.2025).
33. AMD Ryzen(TM) [Электронный ресурс]. — Режим доступа: https://www.amd.com/en/products/processors/desktops/ryzen/7000-series/amd-ryzen-5-7500f.html (дата обращения: 27.05.2025).
34. Windows Developer Blog [Электронный ресурс]. — Режим доступа: https://blogs.windows.com/windowsexperience/2021/06/24/introducing-windows-11/ (дата обращения: 27.05.2025).

ПРИЛОЖЕНИЕ А

Презентация к выпускной квалификационной работе состоит из 19 слайдов.