Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(национальный исследовательский университет)»

На правах рукописи

ЗЫКИН Владимир Сергеевич

МЕТОДЫ И АЛГОРИТМЫ ПОДДЕРЖКИ ЦЕЛОСТНОСТИ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ В ПРИЛОЖЕНИЯХ КЛАССОВ OLAP И OLTP

Специальность 05.13.17 – Теоретические основы информатики

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
ЦЫМБЛЕР Михаил Леонидович,
кандидат физ.-мат. наук, доцент

Челябинск – 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Целостность данных является ключевым понятием в современных реляционных базах данных (БД) [113]. Целостность регламентирует соответствие данных в БД их структуре, логике и всем заданным правилам. Реляционные системы управления базами данных (СУБД) строятся в соответствии с трехуровневой архитектурой *ANSI–SPARC* [30]. *Внутренний* (физический) уровень отвечает за физический способ организации данных. *Промежуточный* (концептуальный) уровень инкапсулирует реляционную схему БД. *Внешний* (пользовательский) уровень показывает, как выглядит БД с точки зрения конечного пользователя и реализуется с помощью представлений. На концептуальном уровне *ссылочная целостность*, наряду с ограничением домена и ограничением сущности, является одним из фундаментальных классов ограничений целостности [9]. Ссылочные ограничения (referential integrity) реализуются в виде зависимостей включения и позволяют сохранить структурную целостность данных, устанавливая логическое соответствие между кортежами отношений БД. В большинстве существующих СУБД, с некоторыми ограничениями (наличие уникального индекса), поддерживается такой вид ограничений.

В настоящее время стали широко использоваться средства автоматизации проектирования схемы БД [40], в которых не регламентируется решение семантических проблем, как это делается на начальных этапах синтетического и декомпозиционного подхода к проектированию [89]. Прежде всего, при объектно-ориентированном (автоматизированном) подходе, не решаются проблемы синонимов и омонимов в списке атрибутов в следствие этого появляются дублированные и противоречивые атрибуты в БД. При объектно-ориентированном подходе формируются нетипизированные зависимости включения, установленные между близкими по смыслу атрибутами, либо

между атрибутами, между которыми существует функциональная зависимость. Ошибки проектирования в итоге нарушают логическую независимость данных, что приводит к проблемам пользователей таких систем БД: значительные финансовые и временные потери при сопровождении и модернизации.

Актуальным аспектом целостности данных является обработка неопределенных значений, которые используются для обозначения факта отсутствия информации. Неопределенные значения ($NULL$ – значения) в технологиях БД являются *актуальной* проблемой с момента существования реляционной модели данных. Свой вклад неопределенности вносят во все виды зависимостей, используемые при проектировании и эксплуатации БД [45]. Это касается и зависимостей включения, а значит, и ссылочной целостности данных. Предложенные ранее решения указанной проблемы основаны на использовании нетипизированных зависимостей включения, что приводит к необходимости перестановок атрибутов, тогда как в технологиях БД атрибуты идентифицируются своим именем, а не позицией в структуре логической записи.

При использовании нетипизированных зависимостей включения возникает необходимость совместной аксиоматизации функциональных зависимостей и зависимостей включения [47]. Алгоритмы поиска полного и неизбыточного набора таких зависимостей включения характеризуются экспоненциальной сложностью [45]. Перечисленные проблемы требуют формализации теории типизированных зависимостей включения, которые допускают неопределенные значения, что влечет за собой упрощение этой системы и приобретение важных свойств: отсутствие взаимодействия с функциональными зависимостями и полиномиальные алгоритмы построения зависимостей включения. Следствием изложенного является то, что проблема построения множества типизированных зависимостей включения является *актуальной*.

Важность ссылочных ограничений целостности в управлении данными подкрепляется современными приложениями, такими как профилирование данных, очистка данных, разрешение объектов и сопоставление схем [67]. Существенную роль ссылочные ограничения целостности играют в технологии формирования многомерных данных [1, 13, 14, 64, 77]. Классический подход построения многомерного представления данных из исходного реляционного представления данных (ROLAP – relational online analytical processing) предполагает иерархические связи на схеме БД («звезда» или «снежинка»). Причем в корне иерархии должно находиться отношение со значениями мер, а в нижних уровнях – отношения, содержащие значения размерностей многомерного представления данных.

Представления (views) рассматриваются на внешнем уровне архитектуры ANSI–SPARC. В современных технологиях БД остается *актуальной* задача поддержки целостности данных при обновлении представлений [56, 57]. В современных БД для обновления представлений программисту необходимо создавать новое приложение (триггер) для каждого отдельного представления. Также остается нерешенной задача обновления представлений, где одной записи в представлении соответствует несколько кортежей в базовых отношениях БД. В соответствии с этим остается *актуальной* задача создания универсального подхода к обновлению представлений, в которых одна обновляемая запись может соответствовать нескольким кортежам в отношениях БД. Обновления представлений осуществляются на основе транзакций к БД, следовательно, их следует рассматривать в рамках таких приложений, как оперативная обработка транзакций (OLTP – Online Transaction Processing)[48].

Степень разработанности темы

Первое фундаментальное исследование ссылочных ограничений целостности, основанных на зависимостях включения, было представлено Казановым М. (Casanova M.), Фейджиным Р. (Fagin R.) и Пападимитроу С. (Papadimitriou C.). Они положили начало теоретическому исследованию зависимостей включения, представили формальное определение нетипизированных зависимостей включения с учетом их взаимодействия с функциональными зависимостями, представили простую аксиоматизацию для зависимостей включения. В своих работах Левин М. (Levene M.) и Лоизу Дж. (Loizou G.) впервые исследовали обобщение зависимостей включения, когда допускается наличие неопределенного значения атрибутов в БД. Левин М. в соавторстве с Винсентом М. (Vincent M.W.) впервые ввели понятие и исследовали нормальные формы зависимостей включения. В последние годы Линк С. (Link S.) с соавторами занимаются исследованием выводимости нетипизированных зависимостей включения с неопределенными значениями, кроме того, набирают популярность исследования условных зависимостей включения Ма С. (Ma S.), Фан В. (Fan W.), Браво Л. (Bravo L.).

Работам, связанным с целостностью данных в классах задач OLAP и OLTP, посвящено большое количество научных работ, известными исследователями в этих областях являются Кузнецов С.Д., Зыкин С.В., Новиков Б.А., Хаямизу Ю. (Hayamizu Y.), Кавамичи Р. (Kawamichi R.), Года К. (Goda K.), Китсурегава М. (Kitsuregawa M.), Левин С. (Levine C.), Ли Ю. (Li Y.).

Исследования, посвященные задаче поддержки целостности данных при выполнении обновлений в представлениях, начаты почти с момента становления реляционной модели данных и продолжаются по сей день. В первых работах, опубликованных в 1981 году, исследователи Банцилхон Ф. (Bancilhon F.), Спиратос Н. (Spyratos N.), Даял У. (Dayal U.), Бернстерн П.А., (Bernstein P.A.) описали проблему обновления представлений, свойства и

условия существования. В своих работах Келлер А. (Keller A.), Лангерак Р. (Langerak R.), Бертосси Л. (Bertossi L.), Салими Б. (Salimi B.) рассматривали частные примеры обновления представлений при решении сторонних задач. В последние годы Масунага Ю. (Masunaga Y.), Нагата Ю. (Nagata Y.), Иши Т. (Ishii T.) представили практическую реализацию обновления представлений в свободной СУБД PostgreSQL. Одной из важных нерешенных прикладных задач остается задача поддержки целостности данных при обновлении многотабличных представлений, снимающая ограничение соответствия одной строки в представлении и одной строки в базовой таблице БД.

Цель и задачи исследования

Целью данной работы является исследование и разработка эффективных методов и алгоритмов поддержки целостности данных на внешнем и концептуальном уровнях архитектуры реляционных баз данных для приложений классов OLAP и OLTP. Для достижения этой цели были поставлены следующие *задачи*:

1. Разработать систему аксиом типизированных зависимостей включения, которая обеспечивает ссылочную целостность при наличии неопределенных значений.
2. Разработать алгоритм построения неизбыточного множества типизированных зависимостей включения с доказательством его корректности и оценкой вычислительной сложности.
3. Разработать общий подход к обновлению многотабличных представлений, обеспечивающий корректную модификацию записи в представлении, которой соответствуют несколько кортежей в хранимых отношениях БД.
4. Реализовать предложенные методы и подходы в виде сопроцессора СУБД для приложений классов OLAP и OLTP.

5. Провести вычислительные эксперименты, подтверждающие эффективность предложенных подходов.

Научная новизна

Научная новизна заключается в том, что впервые была построена полная и непротиворечивая система аксиом для типизированных зависимостей включения, допускающих наличие неопределенных значений. По сравнению с ранее известной аксиоматикой нетипизированных зависимостей включения рассматриваемые в данной работе типизированные зависимости включения устанавливаются только по совпадающему множеству атрибутов, что обеспечивает независимость данных от структуры БД. Разработан оригинальный алгоритм построения типизированных ациклических зависимостей включения. Разработан оригинальный общий подход к обновлению многотабличных представлений на основе коммутативных преобразований данных, который, в отличие от аналогов, позволяет обновлять запись в представлении, которой соответствуют несколько кортежей в базовом отношении.

Теоретическая и практическая значимость

Теоретическая ценность работы заключается в том, что была сформулирована система аксиом для типизированных зависимостей включения. Получено новое доказательство полноты и непротиворечивости указанной системы аксиом, схема которого может быть использована при анализе других видов зависимостей в БД. Доказана корректность и получена оценка вычислительной сложности алгоритма формирования неизбыточного множества типизированных зависимостей включения. Предложен общий подход к обновлению многотабличных представлений, доказана корректность операций.

Практическая значимость работы заключается в реализации сопроцессора реляционной СУБД, обеспечивающего обновление многотабличных

представлений, в которых одной записи в представлении соответствует несколько кортежей в хранимых отношениях БД. Результаты, полученные в диссертационной работе, могут быть использованы для создания инструментальных средств проектирования схем баз данных и сопроцессоров обновления многотабличных представлений для различных коммерческих и свободно распространяемых реляционных СУБД.

Методология и методы исследования

Методологической основой диссертационной работы являются методы математической логики, теория проектирования реляционных БД, теория множеств и реляционная алгебра. Для программной реализации разработанных подходов применялись методы объектно-ориентированного проектирования, язык UML и методы модульного программирования.

Положения, выносимые на защиту

1. Предложена система аксиом типизированных зависимостей включения с неопределенными значениями в реляционных БД и доказана ее полнота и непротиворечивость.
2. Разработан алгоритм построения неизбыточного множества типизированных зависимостей включения в реляционных БД, доказана его корректность и получена оценка вычислительной сложности.
3. Сформулированы и доказаны теоремы о коммутативных преобразованиях для обновления многотабличных представлений в реляционных БД. Разработана архитектура сопроцессора коммутативных преобразований СУБД и реализован сопроцессор СУБД PostgreSQL.
4. Проведены вычислительные эксперименты, подтверждающие эффективность предложенных подходов.

Степень достоверности результатов

Достоверность научных результатов, полученных в работе, подтверждается строгими математическими доказательствами. Теоретические решения подтверждаются вычислительными экспериментами, проведенными в соответствии с общепринятыми стандартами.

Апробация результатов исследования

Основные положения диссертационной работы, разработанная формальная теория, методы, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных научных конференциях:

1. Международная научная конференция «Математика в современном мире» (14–19 августа 2017 г., Новосибирск).
2. The 2th International Conference «Numerical Computations: Theory and Algorithms» (NUMTA2016) (19–25 June 2016, Pizzo Calabro, Calabria, Italy).
3. The 10th IEEE International Scientific and Technical Conference on Dynamics of Systems, Mechanisms and Machines (15–17 November 2016, Omsk, Russia).
4. The VI International Conference «Optimization and Applications» (OPTIMA-2015) (September 27 – October 3, 2015, Petrovac, Montenegro).
5. III Российская молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (24–26 апреля 2013 г., Омск).
6. XVI Байкальская международная школа-семинар «Методы оптимизации и их приложения» (30 июня – 6 июля 2014 г., о. Ольхон, Байкал).

7. IV Международная молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (22–28 апреля 2014 г., Омск).

8. V Международная молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (23–30 апреля 2015 г., Омск).

9. VI Международная молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (22 апреля – 4 мая 2016 г., Омск).

10. VII Международная молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (25 апреля – 4 мая 2017 г., Омск).

11. VIII Международная молодежная научно-практическая конференция «Прикладная математика и фундаментальная информатика» (26 апреля – 4 мая 2018 г., Омск).

Публикации соискателя по теме диссертации

Основные результаты по теме диссертации изложены в следующих научных работах.

Статьи в журналах из перечня ВАК

1. *Зыкин В.С.* Ссылочная целостность данных в корпоративных информационных системах // Информатика и ее применения. 2015. Т. 9, № 3. С. 97–105.

2. *Зыкин В.С.* Инструментальная среда формирования внешних ключей на схеме реляционной базы данных // Омский научный вестник. 2017. № 1. (151). С. 140–143.

3. *Зыкин В.С., Цымблер М.Л.* Обновление многотабличных представлений на основе коммутативных преобразований базы данных // Вестник ЮУрГУ. 2019. Серия: Вычислительная математика и информатика. Т. 8, № 2. С. 92–106.

4. *Зыкин В.С., Зыкин С.В.* Анализ типизированных зависимостей включения с неопределенными значениями // Моделирование и анализ информационных систем. 2017. Т. 24, № 2. С. 155–167.

5. *Зыкин В.С., Зыкин С.В.* Коммутативные преобразования в базе данных при редактировании многотабличных запросов // Информационные технологии. 2018. Т. 24, № 5. С. 330–338.

Статьи в изданиях, индексируемых в SCOPUS и Web of Science

6. *Zykin V., Zykin S.* Analysis of Typed Inclusion Dependences with Null Values // Automatic Control and Computer Sciences. 2018. Vol. 52, Iss. 7, P. 638–646.

7. *Zykin S., Zykin V.* Updates of View in Relational Databases // Proceedings of the 12th International Scientific and Technical Conference “Dynamics of Systems, Mechanisms and Machines”, Dynamics 2018, Omsk, Russia, 13–15 November 2018. Article no. 8601495.

8. *Zykin V.* Automatization of Foreign Keys Construction // Proceedings of the 10th International Scientific and Technical Conference “Dynamics of Systems, Mechanisms and Machines”, Dynamics 2016, Omsk, Russia, 15–17 November 2016. Article no. 7819118.

Свидетельства о государственной регистрации программ для ЭВМ

9. *Зыкин В.С.* Программа для построения неизбыточного набора связей на схеме баз данных: свидетельство о государственной регистрации программ для ЭВМ. № 2018661248 от 04.09.2018.

10. *Зыкин В.С.* Редактор многотабличного представления данных: свидетельство о государственной регистрации программ для ЭВМ. № 2018661249 от 04.09.2018.

Статьи в изданиях, индексируемых в РИНЦ

11. *Зыкин В.С.* Ограничения целостности для неопределенных значений кортежей // Прикладная математика и фундаментальная информатика. 2015. № 2. С. 227–231.

12. *Зыкин В.С.* Сравнительный анализ различных СУБД при редактировании многотабличных представлений данных // Информационный бюллетень Омского научно-образовательного центра ОмГТУ и ИМ СО РАН в области математики и информатики. 2017 № 1 (1). С. 153–154.

Публикации. По теме диссертации опубликовано 10 печатных работ. Работы [1–5] опубликованы в журналах, включенных ВАК в перечень изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора и кандидата наук. Работы [6–8] опубликованы в изданиях, индексируемых в Scopus и Web of Science. В рамках выполнения диссертационной работы получено два свидетельства о государственной регистрации программы для ЭВМ [9–10]. Работы [11–12] включены в перечень изданий, индексируемых в РИНЦ.

Личный вклад. В работе 3 Зыкину В.С. принадлежат разделы 1–4 (обзор работ, коммутативные преобразования реляционных отношений, сопроцессор коммутативных преобразований, экспериментальное исследование, заключение, стр. 94–106), научному руководителю Цымблеру М.Л. принадлежит введение (стр. 92–94). В работе 4 Зыкину В.С. принадлежат разделы 1–4 (обзор результатов, основы формальной теории, минимальное покрытие множества зависимостей и заключение, стр. 155, 157–167), Зыкину С.В. принадлежит введение (стр. 156). В работе 5 Зыкину В.С. принадлежат разделы

1–3 (обзор работ, метод коммутативных преобразований данных, редактирование многотабличных представлений данных и заключение, стр. 332–338), Зыкину С.В. принадлежит введение (стр. 330–332). В работе 6 Зыкину В.С. принадлежат разделы 1–4 (обзор результатов, основы формальной теории, минимальное покрытие множества зависимостей и заключение, стр. 156–167), Зыкину С.В. принадлежит введение (стр. 155). В работе 7 Зыкину В.С. принадлежат разделы 2–5 (обзор работ, метод коммутативных преобразований, обновление многотабличных представлений, заключение, стр. 1–7), Зыкину С.В. принадлежит введение (стр. 1).

Объем и структура работы

Диссертация состоит из введения, четырех глав, заключения и списка литературы. Объем диссертации составляет 137 страниц. Список литературы содержит 119 наименований.

Содержание работы

В первой главе, «Современные методы и средства контроля целостности в реляционных базах данных», проводится обзор научных работ по всем современным направлениям развития теории БД. Особое внимание уделяется теории зависимостей включения, ссылочной целостности и теории обновления представлений. Дается обзор публикаций, наиболее близко относящихся к теме диссертации.

Во второй главе, «Ссылочная целостность отношений на основе теории зависимостей включения», вводится новая система аксиом для типизированных зависимостей включения с неопределенными значениями, доказывается ее полнота и непротиворечивость. Описывается замыкание и минимальное покрытие множества зависимостей включения, Предлагается способ автоматизации построения неизбыточного множества типизированных

зависимостей включения. Приводится способ внедрения разработанной теории в СУБД.

В третьей главе, «Целостность данных многотабличных представлений на основе коммутативных преобразований данных», предлагается новый подход к обновлению многотабличных представлений, основанный на аппарате коммутативных преобразований БД. Приводятся формулы, реализующие операции обновления многотабличных представлений в терминах реляционной алгебры. Формулируются и доказываются теоремы о корректности преобразований, выполняемых в соответствии с предложенными формулами. Описана архитектура сопроцессора СУБД, выполняющего обновление представлений на основе коммутативных преобразований.

В четвертой главе, «Программная реализация и экспериментальные исследования», описывается реализация подхода к обновлению многотабличных представлений на основе коммутативных преобразований данных применительно к свободной СУБД PostgreSQL. Приводятся результаты вычислительных экспериментов, показывающих эффективность разработанных подходов в приложениях классов OLAP и OLTP.

В заключении перечисляются результаты, полученные в итоге выполнения диссертационного исследования, проводится сравнение с наиболее близкими работами по данной тематике, даются рекомендации по использованию разработанного подхода и программного обеспечения, рассматриваются направления дальнейших исследований.

ГЛАВА 1. СОВРЕМЕННЫЕ МЕТОДЫ И СРЕДСТВА КОНТРОЛЯ ЦЕЛОСТНОСТИ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

В данной главе рассматриваются тенденции развития теоретических аспектов ссылочной целостности и дается обзор научных исследований в области современных технологий БД. Анализируются публикации, наиболее близко относящиеся к теме диссертации.

1.1 Понятие и аспекты целостности данных

«Понятие согласованности, или целостности данных является ключевым понятием БД» [113]. «Целостность БД (database integrity) – соответствие имеющейся в БД информации ее внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние БД, называется ограничением целостности (integrity constraint)». Под *ограничением целостности* будем понимать «свойство БД, определяемое способностью СУБД защищать компоненты и связи БД от искажения в результате некорректных операций и сбоев технических средств» [88].

Функции поддержки ограничений целостности в технологии БД выполняет СУБД. Результат любой операции (вставка, удаление, обновление) будет откатан СУБД, если операция нарушает ограничение целостности.

Целостность БД на концептуальном уровне архитектуры ANSI–SPARC осуществляется при помощи следующих ограничений целостности [113]:

1) ограничение домена;
2) ограничение сущности;
3) ссылочные ограничения.

Представление (view) – это виртуальное (логическое) отношение БД, которое является синонимом запроса к хранимым отношениям БД. Механизм представлений позволяет скрывать детали концептуальной структуры БД от конечных пользователей.

Целостность данных на внешнем уровне архитектуры ANSI–SPARC подразумевает соответствие данных в хранимых таблицах БД информации, предоставленной пользователю в представлениях.

В данной диссертационной работе исследуется ссылочная целостность данных касательно концептуального уровня, а также целостность данных в хранимых таблицах БД при обновлении представлений на внешнем уровне архитектуры ANSI–SPARC.

Ограничение домена

При определении структуры отношения БД могут быть заданы ограничения на допустимые значения в столбцах. Определение типа атрибута в отношении задает базовое ограничение, которое контролируется СУБД. В указанном столбце не может появиться значение, противоречащее выбранному типу, например, символьная строка в столбце, для которого указан тип «Дата». Кроме того, на значения атрибутов в столбце отношения могут быть дополнительные ограничения, например, дата должна быть задана в определенном интервале. Попытка ввести значение этого атрибута, лежащее за пределами указанного интервала, будет откатана СУБД.

Целостность сущностей

В каждом отношении должен быть задан первичный ключ, который имеет уникальное непустое значение в каждой строке отношения. Основанием для определения первичного ключа является множество функциональ-

ных зависимостей, формируемых проектировщиком БД [89]. Определенному таким образом набору атрибутов отношения ставится в соответствие свойство PRIMARY KEY. В пределах отношения это свойство может указываться только один раз. Другим способом гарантировать уникальность значений подмножества атрибутов для потенциальных ключей отношения является присвоение этому подмножеству свойства UNIQUE.

Ссылочная целостность

Ссылочные ограничения целостности (Referential Integrity Constraint) – класс ограничений целостности, при котором кортеж с некоторым значением данного набора атрибутов может быть включен в отношение тогда и только тогда, когда это значение является актуальным значением первичного ключа указанного другого отношения [112]. Для реализации данного бизнес-правила в СУБД имеется аппарат внешних ключей (FOREIGN KEY). При этом определяется главное отношение и подчиненное отношение: в подчиненном отношении не может быть строк, которым нет соответствующей строки в главном отношении. СУБД произведет откат операции вставки строки в подчиненное отношение, если в главном отношении нет соответствующего кортежа, и операция удаления строки в главном отношении будет отвергнута, если в подчиненном отношении имеется соответствующая связанная строка (запрет висячих ссылок).

1.2 Целостность отношений баз данных

Ссылочные ограничения целостности являются одним из основных видов ограничений, которые позволяют сохранить структурную целостность БД. С другой стороны, неопределенные значения ($NULL$) стали *актуальной* проблемой с момента создания реляционной модели данных. Влияние неопределенностей сказывается на всех видах зависимостей, используемых при проектировании и эксплуатации БД. В полной мере это относится и к

зависимостям включения, которые являются теоретической основой ссылочной целостности данных.

Формальное определение зависимостей включения приведено в работе [9]: пусть $U = \{A_1, A_2, \dots, A_n\}$ – множество атрибутов, определенных в БД, $[R_i]$ – множество атрибутов, на которых определено отношение R_i, $[R_i] \subseteq U$, $1 \le i \le k$, $\mathfrak{R} = (R_1, R_2, \dots, R_k)$ – БД, $S = \{[R_1], [R_2], \dots, [R_k]\}$ – схема БД.

Определение 1.1. Пусть $[R_i]$ и $\left[R_j\right]$ – схемы отношений (не обязательно различные), $V \subseteq [R_i]$ и $W \subseteq \left[R_j\right]$, $|V| = |W|$, тогда соотношение $R_i\,[V] \subseteq R_j\,[W]$ называется *зависимостью включения*.

В определении 1.1 $|V|$ – мощность множества V, $R_i\,[V] = \boldsymbol{\pi}_V\,(R_i)$ – проекция отношения R_i по атрибутам V. Зависимость включения считается *типизированной*, если $V = W$, в противном случае – *нетипизированной*.

Далее в работе [9] представлена система аксиом зависимостей включения:

- **IND1** (рефлексивность): $R_i[X] \subseteq R_i[X]$, если X последовательность отдельных атрибутов R_i.
- **IND2** (проецирование и перестановка): если $R_i\,[A_1, \dots, A_m] \subseteq R_j\,[B_1, \dots, B_m]$, тогда $R_i\left[A_{i_1}, \dots, A_{i_q}\right] \subseteq R_j\left[B_{i_1}, \dots, B_{i_q}\right]$ для каждой последовательности $i_1, \dots, i_q$ различных целочисленных значений из множества $\{1, \dots, m\}$.
- **IND3** (транзитивность): если $R_i\,[X] \subseteq R_j\,[Y]$ и $R_j\,[Y] \subseteq R_l\,[Z]$, тогда выполнено $R_i\,[X] \subseteq R_l\,[Z]$.

Относительно системы IND1–IND3 в [9] представлено доказательство полноты и показано, что система является надежной (непротиворечивой). Система аксиом является *непротиворечивой*, если из нее нельзя вывести два взаимоисключающих утверждения. Систему аксиом будем называть *полной*, если отсутствует любая другая аксиома, которая не выводима из системы аксиом.

Проблеме исследования зависимостей включения до сих пор уделяется внимание со стороны исследователей, поскольку, с одной стороны, остаются нерешенными некоторые теоретические проблемы. С другой стороны, практика использования БД формулирует новые требования к ссылочным ограничениям целостности.

В современных технологиях БД ссылочные ограничения целостности играют важную роль в таких областях как: профилирование и очистка данных [67], в приложениях классов OLAP и OLTP [13], в многомерных данных реляционной [14] и объектно-ориентированной модели [1], а также при их обработке [64, 77].

Попытки решения указанной проблемы содержат неточности, как в постановке задачи, так и в самом ее решении. К постановочным ошибкам можно отнести использование в определении нетипизированных зависимостей включения, что приводит к перестановкам атрибутов, хотя в технологиях БД атрибуты идентифицируются по имени, а не по их позиции. Кроме того, связывание зависимостью включения разнородных, пусть даже однотипных, атрибутов является признаком потерянной функциональной зависимости и приводит к взаимодействию нетривиальных зависимостей включения и функциональных зависимостей. Зависимости включения должны определять количественное соотнесение объектов друг с другом, а не значений атрибутов. Неточности в решении указанной проблемы содержатся в форму-

лировках аксиом и доказательстве их свойств, в том числе полноты. Функциональные зависимости и зависимости включения, устанавливаемые по разнородным атрибутам в конечном счете, *нарушают принцип независимости данных*.

Классический способ проектирования схемы БД [89, 113, 115] основан на зависимостях: функциональных, многозначных и соединения. При этом сущности (отношения) БД формируются в процессе проектирования, что позволяет реализовать принцип независимости данных и, как следствие, устойчивость проекта БД при последующей модернизации.

Альтернативным подходом к созданию проекта БД является объектно-ориентированный подход. В этом случае сначала создаются сущности (объекты) БД. Корректность созданных отношений остается полностью за проектировщиком. Зачастую объекты дублируют структуру документооборота предприятия, что приводит к разрушению структуры БД при необходимости ее модернизации. Затем проектировщик вручную создает связи между объектами. Типичной ошибкой при таком подходе является «нагруженность» связей семантикой приложения, то есть в связях «прячутся» объекты БД. Несмотря на все недостатки объектно-ориентированного подхода, он получил достаточно широкое распространение на практике за счет двух своих преимуществ: наглядности проектирования и низким уровнем требований к квалификации проектировщика.

В качестве примера такого инструментария можно привести ERwin Process Modeler [2]. При преобразовании ER-диаграммы на схему БД используется механизм автоматического формирования связей. Правила преобразования при этом являются эвристическими с множеством исключений и нереализуемых ситуаций, причина которых в неоднозначной интерпретации сущностей и связей. Следовательно, говорить об автоматизации этого процесса нет смысла. Признаком хорошей технологии является повторяемость резуль-

тата: два различных человека при моделировании одних и тех же данных получают одинаковый результат. Для синтетического подхода формирования схемы БД на основе неизбыточного множества зависимостей это является очевидным, но требует от проектировщика знаний по теории проектирования БД. При объектно-ориентированном проектировании результаты в приложениях могут значительно различаться, что говорит о нарушении принципа независимости данных.

Рассмотрим два типовых примера использования нетипизированных зависимостей включения. Первый пример представлен в научной статье Левена и Винсента (Levene, Vincent) [47]. В примере 3 атрибута: H – руководитель отдела, L – лектор, D – отдел; два отношения: $R_1[HD]$ и $R_2[LD]$, зависимости $L \to D$ и $R_1[HD] \subseteq R_2[LD]$. Этот пример представлен для демонстрации взаимодействия функциональных зависимостей и зависимостей включения. Так, из совокупности двух рассмотренных зависимостей логически следует функциональная зависимость $H \to D$. В этом примере перед построением декомпозиции не были решены семантические проблемы. Атрибуты H и L однородны, общее между ними то, что это сотрудники, различие между ними – должности. Заменим атрибуты H и L атрибутами P – должность сотрудника, F – фамилия сотрудника. У перечисленных атрибутов нет однозначного идентификатора, так как однофамильцы могут работать в разных отделах. Поэтому дополняем атрибут N – номер сотрудника. По правилам построения нормальных форм должно существовать одно отношение $R[NFPD]$, построенное с использованием зависимости $N \to FPD$, где подчеркнут ключевой атрибут отношения. Если обобщить этот пример, то предположение о взаимодействии типизированных зависимостей включения с функциональными зависимостями противоречит минимальному покрытию множества функциональных зависимостей, которое используется при построении нормальных форм.

Далее рассмотрим другой практический пример использования нетипизированных зависимостей включения, который был использован разработчиками MySQL [27]. Очевидно, что при формировании схемы использовались в основном эвристики, а зависимости на данных имели второстепенное значение, или игнорировались вовсе. Поэтому схема содержит множество неточностей. Рассмотрим одну из них, касающуюся содержания данной работы. Отношение (таблица) «film» имеет в своем составе атрибуты: «language_id» и «original_language_id». Для этих атрибутов установлена ссылочная целостность с отношением «language» по атрибуту «language_id». В результате имеем нетипизированную зависимость включения:

$$film[original\_language\_id] \subseteq language[language\_id].$$

Все фильмы имеют значение атрибута «язык оригинала», однако только некоторые из них имеют перевод. Могут быть еще атрибуты: «автор перевода», «дата выпуска перевода» и т.д. Рассмотренный пример наглядно показывает, как данные о фильмах зависят от структуры БД, следовательно, при дальнейшем развитии системы возникнут проблемы работы прикладных программ с БД.

Для удаления зависимости данных от структуры БД, необходимо преобразовать исходную схему БД, добавив отношение, которое нужно связать с отношением «film». На схеме должно существовать отдельное отношение «translate» с типизированными зависимостями включения:

$$translate[language\_id] \subsetneq language[language\_id]$$

и

$$translate[film\_id] \subsetneq film[film\_id],$$

а нетипизированная зависимость должна быть удалена вместе с соответствующим атрибутом. Отметим, что в основе данного преобразования лежит

сравнение семантики однородных, но разноименных атрибутов: original_language_id и language_id совпадают в том, что это идентификаторы языка озвучивания фильма. Различие в том, что один из них подвергся трансляции, а другой нет.

Рассмотрим другой пример БД, на котором можно наглядно продемонстрировать актуальность использования неопределенных значений. На этом же примере покажем, в чем заключаются недостатки существующей теории зависимостей включения.

Рассмотрим фрагмент схемы БД кинотеатра:

R_1 = *Фильмы* (№ фильма, Наименование фильма, Жанр фильма);

R_2 = *Расписание сеансов* (Дата сеанса, Время сеанса, № зала, № фильма);

R_3 = *Билеты* (№ билета, Дата сеанса, Время сеанса, № зала, № ряда, № места).

В рассмотренном примере на схеме должны быть установлены следующие ссылочные ограничения целостности: нельзя в расписании назначить сеанс, если неизвестен фильм. Однако, возможно продать билет со свободной (неопределенной) датой посещения, и/или свободным временем, и/или неопределенным номером зала и т.д. Это означает, что определенным значениям в отношении R_2 могут быть поставлены в соответствие неопределенные значения ($NULL$) в R_3. И если появится необходимость заменить неопределенные значения в R_3 на определенные, то выбор может быть сделан только из соответствующих определенных значений в R_2. Однако, в существующей теории зависимостей включения неопределенным значениям в R_2 могут соответствовать определенные значения в R_3. На практике это означает неопределенные значения в первичном ключе в таблице R_2.

1.3 Целостность представлений баз данных

Под *представлением* (view) БД будем понимать виртуальную таблицу, которая определяется на основе хранимых таблиц БД и/или других представлений [112]. Механизм представлений играет важную роль как средство структурирования информации для обеспечения потребностей конкретных пользователей. Доступ к чтению через представления легко реализуется средствами языков запросов, тогда как обновление данных в БД с помощью представлений является довольно сложной проблемой. Для разработчиков программного обеспечения СУБД обновление данных с помощью представлений является актуальной проблемой, поскольку соответствующий инструментарий существенно сократит время разработки типовых приложений.

Задача обновления многотабличных представлений (view) является актуальной при создании приложений для работы с реляционной БД [41]. Данной проблеме до сих пор уделяется большое внимание в научных работах. Приложение должно работать с представлением данных – таблицей, сформированной из множества отношений БД с использованием запросов типа «соединение-селекция-проекция». Требованием к таким приложениям является реализация функций обновлений данных: вставка, удаление и обновление записи в представлении. Для обеспечения корректности данных необходимо, чтобы преобразование удовлетворяло условию коммутативности. Для приложений класса OLAP (On-line Analytical Processing) применение инструмента обновления представлений ограничивается вставкой новых записей в хранилище данных. Обновления представлений для задач OLTP (On-line Transaction Processing) можно осуществлять при рассмотрении различных транзакций в соответствии со сценарием конкретного приложения.

Ограничения на обновления представлений данных присутствуют, начиная с первого стандарта языка SQL (Structured Query Language), суть ко-

торых сводится [21] к необходимости СУБД «для каждой строки представления найти соответствующую строку в исходном отношении, а для каждого обновляемого столбца представления – соответствующий столбец в исходном отношении». При этом, «если представление пропускает столбец исходного отношения с ограничением NOT NULL без спецификации DEFAULT, то нельзя выполнить вставку нового кортежа в отношение посредством данного представления. Причина в том, что в этом случае нет способа указать значение данных для пропущенного столбца».

Большинство коммерческих СУБД лишь частично ослабляют базовые ограничения стандарта SQL [3, 19, 62]. Например, выделенные столбцы в представлении могут быть предназначены только для просмотра, попытка обновления таких столбцов блокируется, тогда как значения в других столбцах можно изменять в пределах заданных ограничений на соответствующие столбцы в исходном отношении.

Основная причина перечисленных и иных проблем: должно быть установлено соответствие записей обновляемого представления и кортежей отношений БД [21], что не всегда удается сделать.

Для решения проблем обновления представлений СУБД используют механизм триггеров. Однако такое решение имеет ряд недостатков: триггеры необходимо разрабатывать для каждой отдельной БД и для каждого отдельного случая бизнес-правила. Триггеры являются источником накладных расходов СУБД, поскольку требуют блокировки различных ресурсов, проверки наступления события триггера, поддержки временных таблиц (например, таблицы INSERTED и DELETED в СУБД MS SQL Server используются для проверки результатов изменений данных и установки условий срабатывания триггеров [23]) и др. Кроме того, при наличии нескольких триггеров, относящихся к одному отношению (представлению) и ассоциированных с одним событием, последовательность их запуска не детерминирована [104].

1.4 Обзор работ по теме диссертации

В данном разделе делается обзор работ, наиболее близко относящихся к теме диссертации.

1.4.1 Обзор результатов по ссылочной целостности данных

В предыдущем разделе было указано, что зависимости включения являются теоретической основой ссылочных ограничений целостности. В данной диссертационной работе рассмотрены типизированные зависимости включения, которые, по мнению автора, наиболее приемлемы при классическом подходе к проектированию БД [113, 115, 118]. Количественное соотнесение значений атрибутов друг с другом, прежде всего, определяется функциональными зависимостями. Тогда как задача зависимостей включения определять количественное соотнесение объектов друг с другом. Использование нетипизированных зависимостей включения приводит к смешению этих двух базовых видов зависимостей и, как следствие, к проблемам при проектировании схемы БД. Причем, эти проблемы сказываются не только в теории, но и на практике, когда объект БД подменяется связью на схеме.

Работу Казановы (Casanova), Фэйджина (Fagin) и Пападимитроу (Papadimitriou) [9] можно называть пионерской в области зависимостей включения, здесь исследуются только нетипизированные зависимости включения, приводится и доказывается система аксиом. Во всех остальных работах, посвященных зависимостям включения, доказательство полноты системы аксиом сводится к ссылке на работу [9]. В дополнение к этой проблеме в работе [9] возникают вопросы при анализе доказательства полноты системы аксиом **IND1–IND3**. При доказательстве условия $\Sigma \vDash \sigma \Rightarrow \Sigma \vdash \sigma$: если зависимость σ выполнима (логически следует из Σ), то σ выводима из Σ с использованием аксиом **IND1–IND3**, рассматривается правило (*Rule*) формирования представления БД, которое удовлетворяет зависимостям Σ.

Затем показано, что если в БД выполнена зависимость σ, то она выводима из Σ. При такой схеме доказательства необходимо показать выводимость σ для любого состояния БД, а не только единственного состояния, построенного по правилу. Несомненно, можно показать выводимость σ для любого состояния БД, но это в работе [9] *не сделано*.

Разносторонний анализ нетипизированных зависимостей включения показал, что полная аксиоматизация существует по отдельности для зависимостей включения и функциональных зависимостей, тогда как совместно для этих зависимостей полная *аксиоматизация отсутствует* [9, 12, 20]. В частных случаях, в том числе для одноместных зависимостей включения и произвольных функциональных зависимостей, существует полная аксиоматизация [15, 36].

Продолжением проблемы взаимодействия зависимостей является построение соответствующих нормальных форм (IDNF), основанных на нормальной форме Бойса–Кодда и с ограничением в виде зависимостей включения для ациклических схем БД [47]. Впервые условия ацикличности схем БД были исследованы в [6]. Структурная интерпретация зависимостей включения в виде графа представлена в [58]. Хотя исследуются отличные от [6] условия ацикличности, в работе [102] показана связь между ними.

Отсутствие полной аксиоматизации говорит о том, что в общем случае *нельзя определить выводимость* той или иной зависимости. Как следствие, процедура построения нормальных форм в общем случае *не имеет решения*. Усугубляет ситуацию использование нетипизированных зависимостей включения. Ссылочная целостность между двумя (и более) неоднородными атрибутами является *некорректной*. Формально это нетривиальная функциональная зависимость, которая могла быть реализована при проектировании в структуре логических записей на схеме БД, а не в виде ссылочной целостно-

сти. Необходимость использования нетипизированных зависимостей включения в этом случае отпадает. С другой стороны, типизированным зависимостям включения соответствуют тривиальные функциональные зависимости, которые при формировании структуры логических записей не используются. Как показано в работе [47], зависимости включения не взаимодействуют с ациклическими функциональными зависимостями в рамках нормальной формы Бойса–Кодда, и могут рассматриваться отдельно друг от друга.

Нетипизированные зависимости включения возникают на атрибутах, которые являются синонимами и фактически являются *различными по семантике атрибутами*. При декомпозиционном и синтетическом подходе к проектированию БД список атрибутов нормализуется, во время чего удаляются атрибуты синонимы и омонимы. При проектировании БД с использованием объектного подхода, этап декомпозиции *пропускается* и как следствие на схеме БД появляются нетипизированные связи между синонимами. Если связь является нетипизированной и атрибуты, по которым устанавливается связь являются не синонимами (имеют различную область определения), то между этими атрибутами можно установить нетривиальную функциональную зависимость. В случае, когда атрибуты являются синонимами, тогда между ними можно установить тривиальную зависимость включения, которая не взаимодействует с нетипизированными зависимостями включения. В случае, когда атрибуты являются омонимами, возможен случай установления типизированной связи, хотя атрибуты являются *различными по семантике*.

Значительная часть работ посвящена разработке и исследованию алгоритмов поиска зависимостей включения. В работе [58] Мисаоуи (Missaoui) представил полиномиальный по времени алгоритм для поиска избыточных зависимостей на основе графического теоретического подхода. Представленный алгоритм является полным аналогом алгоритма поиска избыточных

функциональных зависимостей. В работах [8, 35, 54, 102] представлены различные алгоритмы поиска зависимостей включения, основанные на использовании свойств на схемы БД, анализе данных и обработке входящих запросов. Все эти алгоритмы *не гарантируют корректность* и *полноту* обнаружения зависимостей включения, но позволяют частично автоматизировать этот процесс.

В последнее время наблюдается повышенный интерес к условным зависимостям включения [53]. Идея заключается в том, что зависимости включения удовлетворяют не все кортежи отношения, а только их часть, удовлетворяющая некоторому условию. Этот вид зависимостей хорошо описывается существующей формальной теорией [9]. Однако на практике искать и поддерживать такие зависимости довольно *сложно*. Для такого вида зависимостей может быть использован другой подход, основанный на областях определения зависимостей [106].

Полная автоматизация построения зависимостей включения в общем виде *нереализуема*, поскольку они отражают специфику бизнес-правил в конкретной прикладной области. Однако, способствовать их построению позволяют алгоритмы, которые определяют полноту условий за счет определения пороговых значений качества [5, 24].

В настоящее время активно развивается направление анализа зависимостей включения. Для улучшения и автоматизации проверки и диагностики соблюдения правил на основе описания семантики данных (бизнес-данных) предлагаются структуры и алгоритмы для обнаружения возможных нарушений ограничений бизнес-данных, в том числе при модификации схемы БД [24, 74, 22, 51]. Это подчеркивает *актуальность* проводимых исследований в настоящее время и в будущем.

В данной работе рассматриваются основы формальной теории для типизированных зависимостей включения. Хотя такие зависимости считаются

частным случаем нетипизированных зависимостей включения, система аксиом получилась иной: во второй аксиоме отсутствует необходимость использования перестановок. Сами по себе перестановки в системе **IND1-IND3** являются *искусственными*. Действительно, в технологиях БД атрибуты идентифицируются по имени, а не по расположению. Тогда как перестановки фиксируют расположение атрибута. Вместо перестановок можно использовать переименование атрибутов и т.п., что лишний раз подтверждает искусственную природу нетипизированных зависимостей включения. Далее будет предложено правило преобразования нетипизированных зависимостей включения к типизированному виду.

Наличие неопределенных значений в БД является неизбежным. Поэтому формальная теория должна учитывать их при проектировании. В работах [45, 44, 46] рассматривается проблема выводимости совместно для функциональных зависимостей и зависимостей включения, допускающих наличие неопределенных значений. Как и в случае отсутствия неопределенных значений удается построить полную и непротиворечивую аксиоматику *только в частных случаях*. Основным препятствием для получения более общих результатов является взаимодействие функциональных зависимостей и нетипизированных зависимостей включения. В работах [39, 38] рассматриваются простые и частичные нетипизированные зависимости включения с неопределенными значениями, имеющие место в стандарте языка SQL. Представлены две системы аксиом, содержащие аксиомы с перестановками атрибутов. Для обеих систем аксиом утверждается наличие полноты со ссылкой на работу [45], однако в работе [45] при утверждении полноты системы аксиом ссылка идет уже на упомянутую работу [9], в которой при доказательстве полноты рассматриваются только *частный случай*. В большинстве рассмотренных работ предполагается, что зависимости включения являются нециклическими.

В рассмотренных работах [38, 39, 44, 45, 46] установление зависимостей включения с неопределенными значениями рассматривается не в общем виде, а с помощью наглядных примеров, в предлагаемой диссертационной работе дается четкое формальное определение зависимости включения с неопределенными значениями. В работах [44, 45, 46] допускается формальное установление зависимости включения от внешнего отношения к главному даже в случае, когда определенному значению во внешней таблице соответствует неопределенное значение в главной, хотя такой случай на практике соответствует *NULL значению в первичном ключе*. Кроме того, в таком случае теряется свойство транзитивности зависимостей включения, и система аксиом принимает абсолютно другой вид: правила становятся аксиомами, а аксиома транзитивности *исчезает*. Хотя в работе [9] указывается, что транзитивность является одной из основных аксиом.

Поиск зависимостей включения основывается на анализе уже заполненной БД, что является *трудоемкой задачей*. Так в работе Марчи (Marchi) с соавторами [54] на основе заранее обнаруженной одноместной зависимости включения используется интеллектуальный алгоритм для увеличения арности зависимости: генерация кандидата на зависимость включения арности $i + 1$ из существующей зависимости арности i, $(i > 0)$. Эксперименты проводились на синтетических БД и показали приемлемые результаты.

В работе [5] Баукман (Bauckmsnn) и его соавторы на основе введенного обобщения условных зависимостей включения разработали эффективные алгоритмы, которые определяют условия покрытия и полноты обнаружения зависимостей в соответствии с заданными порогами качества. Предложенные алгоритмы автоматически определяют не только значения условия, но и атрибуты условия.

В работе [24] Гомез-Лопез (Gomez-Lopez) с соавторами исследуют данные в бизнес-процессах, которые могут быть изменены и обновлены различными видами деятельности в любое время. Данные могут быть связаны с элементами потока или сохраненными данными и должны соответствовать правилам, которые определяют поведение компании. Для улучшения и автоматизации проверки и диагностики соответствия правилам предложена структура, в которой автоматически анализируются переменные потока данных и хранимые данные. Кроме того, предложен язык описания ограничений, который позволяет менеджеру создавать ограничения на данные без поддержки эксперта по информационным технологиям.

При условии, что в БД может быть нарушена ссылочная целостность и отношения могут быть денормализованы, в работе [63] Ордонез (Ordonez) и Гарциа-Гарциа (Gracia-Gracia) предлагается набор показателей качества, определенных на нескольких уровнях детализации, включая БД, отношения, атрибуты. Показатели качества измеряют ссылочную полноту и согласованность и эффективно вычисляются стандартными SQL-запросами. Проведенные эксперименты на реальных и синтетических БД показали, что предложенный подход может помочь в обнаружении и объяснении ссылочных ошибок.

В работе [49] Линк (Link) и Мемари (Memari) оценивают стоимость проверки ссылочной целостности для различных размеров наборов данных, структур индексирования и степеней согласованности данных. В результате предложен подход, который повышает эффективность проверки частичных зависимостей, в том числе, зависимостей включения.

При отсутствии в описании данных первичных и внешних ключей в работе [60] Мотл (Motl) и Кордик (Kordik) предлагают автоматически определять основные и внешние ограничения ключа из метаданных. Для этого

использовалась F-мера, равная значению 0,87, для идентификации ограничений внешнего ключа. Предлагаемый метод значительно превосходит методы, связанные со временем выполнения, описанные в литературе. Для быстрого определения связей предсказательная модель обучалась только по метаданным о данных, доступных с API-интерфейсом Java Database Connectivity (JDBC). Этот подход имеет следующие свойства: 1) быстрый и масштабируемый; 2) не требуется информация об источнике БД; 3) не зависит от качества данных. Проблема идентификации связей была разложена на две подзадачи: идентификация первичных ключей и идентификация внешних ключей. В результате обработки метаданных определяются ограничения, обладающие наибольшей вероятностью.

В работе 2017 года [66] Покорный (Pokorny) и его соавторы обсуждают проблемы ссылочных ограничений целостности для графических БД. При этом сравнивается концептуальная схема графических данных с соответствующей ей схемой БД. С использованием средств описания данных рассматриваются ссылочные ограничения БД и ссылочные ограничения графических данных. С использованием графической БД Neo4j, ее возможностей формирования схемы БД и ссылочных ограничений целостности получено расширение этих возможностей с помощью новых конструкций в Neo4j DDL, включая разработку прототипа, реализацию и эксперименты.

В соответствии с произведенным обзором работ по ссылочной целостности остается *актуальной* задача построения новой теории типизированных зависимостей включения и доказательства ее полноты и непротиворечивости. Существенным отличием данной работы является установление ссылочных ограничений целостности, основанных на типизированных зависимостях включения, причем свойства таких зависимостей определяются так, что

сохраняется свойство транзитивности при наличии неопределенных значений. Также устраняются проблемы, связанные с использованием нетипизированных зависимостей включения.

1.4.2 Поддержка целостности данных при обновлении многотабличных представлений

Исследования, посвященные задаче корректного обновления представлений, были начаты с момента становления реляционной модели данных и продолжаются по сей день. Статья [4] была одной из первых в данной тематике. В ней Банцилхон (Bancilhon) и Спиратос (Spyratos) исследуют свойства отображений состояний БД в состояния представлений, основанных на морфизме. Представлена схема коммутативности композиции преобразований, исследования сопровождаются примерами. Однако, отсутствуют какие-либо алгоритмы, соответствующие преобразованиям, и метод «константного комплемента», предложенный Bancilhon и Spyratos, слишком ограничительный, чтобы быть полезным для создания технологии.

В статье [17] Даял (Dayal) и Бернстейн (Bernstein) рассмотрена проблема автоматического перевода обновлений пользовательских представлений в соответствующие обновления исходной БД. Прежде всего были исследованы свойства корректности таких преобразований и условия их существования. Условия были получены сначала с точки зрения расширения схемы и представления, а затем синтаксически с использованием функциональных зависимостей и ключей. Для определения операций преобразования БД были определены два графа: граф-трассировка и граф зависимостей представления. Проведенные исследования показали, что точные и уникальные переводы обновлений могут быть возможны только в довольно жестких условиях. Необходимость построения графов и условия корректности в виде ис-

числения предикатов делают затруднительным использование этого материала для формирования технологии. Кроме того, преобразования, описанные в статье, современные СУБД частично делают автоматически.

В работе [55] Масунага (Masunaga) предпринял попытку решения данной проблемы в общем виде: допускается обновление кортежей во всех отношениях представления и работа с комбинациями представлений. Для решения проблемы неоднозначности интерпретации операций преобразования предлагается ряд эвристических критериев и правил, которые в общем случае не гарантируют корректный результат. В крайнем случае, предполагается вмешательство пользователя в процесс обновления БД.

В работе [16] Космадакис (Cosmadakis) и Пападемитроу (Papadimitriou) рассматривают обратную задачу – обновление представлений данных. В общем случае обновление представлений неоднозначно и требует дополнительное представление, которое содержит всю информацию, опущенную из данного представления, и остается постоянным во время преобразований. Изучаются некоторые вычислительные проблемы, связанные с применением этой общей методологии в контексте реляционных БД. Показано, что поиск минимального дополнения к данному представлению является NP-полной задачей. Корректность преобразований проверяется тестированием, исследуется вычислительная сложность тестирования.

В статье [119] Филлиповичем рассмотрена проблема перевода обновлений представлений в БД, где БД и представление моделируются как абстракции данных. Абстракция данных состоит из набора состояний и набора примитивных операторов обновления, представляющих функции перехода состояния. Показано, как сложные программы обновления могут быть созданы из примитивных операторов обновления, и как программы обновления версий транслируются в программы обновления БД. Исследованы свойства эквивалентности и сериализуемости переходов между состояниями. В работе

определено понятие дополнения и показано влияние постоянного дополнения на изменения в представлении.

В статье [41] Лангерак (Langerak) изучает проблему трансляции обновления представления для реляционной модели данных, в которой базовые отношения могут содержать значения *NULL*. Предполагается наличие репрезентативного экземпляра, который считается правильным представлением всех данных в БД; класс изучаемых представлений состоит из полных проекций репрезентативного экземпляра. Рассматриваются только отдельные схемы БД, то есть схемы, для которых глобальная согласованность подразумевается локальной согласованностью. Условие независимости отношений является сильно ограничительным. Связывание отношений ссылочной целостностью делает их зависимыми. Кроме того, для обеспечения корректности преобразования предполагается формирование расширения обновляемого отношения до размеров универсума (отношения, определенного на всем множестве атрибутов), что само по себе является препятствием для практического использования такого подхода.

В работе [42] Лечтенбёргером (*Lechtenbörger*) продолжены исследования свойств обновления представлений, основанных на "constant complement approach". Получены интересные теоретические результаты, в том числе, применительно к множеству внешних схем. Однако, результаты пока далеки от практического использования.

Наиболее близка к тематике данной работы статья Кёллера (Keller) [37], в которой обсуждается проблема корректности формирования представлений, основанных на реляционных операциях: соединение, селекция и проекция. Автором рассмотрена схема межмодельных преобразований, основанная на условиях коммутативности. Также рассмотрены пять критериев правильности выполненных преобразований, представлены варианты алгоритмов выполнения операций вставки, удаления и обновления записей в

алгоритмах. К недостаткам работы [37] можно отнести отсутствие модели для описания представлений, что является препятствием к созданию технологической цепочки преобразований. Рассмотренные примеры с командами SQL не заменяют отсутствующую модель. Кроме того, для обеспечения корректности преобразования требуется наличие ключевых атрибутов в представлениях. В реальных БД ключами чаще всего являются автоинкрементные атрибуты, в которых хранятся номера соответствующих объектов. Значения таких атрибутов недоступны для обновления и не обладают какой-либо семантикой. Следовательно, работа пользователей с представлениями будет существенно затруднена.

В работе [41] Лангерак (Langerak) исследует задачу трансляции обновления представления с неопределенными значениями в контексте реляционных БД. Считается, что в программе присутствует так называемый репрезентативный экземпляр, который считается полностью корректным. Представления являются проекцией этого экземпляра. В данной работе принимаются к рассмотрению только частные случаи схем БД, где глобальная согласованность подразумевает локальную согласованность. Также довольно жесткие ограничения накладываются на условия независимости отношений. К примеру, в таблице фактов присутствуют внешние ключи практически ко всем отношениям–справочникам, следовательно, все рассматриваемые отношения являются зависимыми. В дополнение ко всему для выполнения корректных операций преобразования данных предлагается использовать временное универсальное представление, которое содержит все атрибуты БД. Данное условие является достаточно жестким ограничением при практическом использовании ввиду необходимости хранить в оперативной памяти информацию со всей БД.

В 2003 г. Лечтенбёргер в статье [42] представил подход «постоянного дополнения» (constant complement approach), развивающий идеи работы [4].

Данный подход предоставляет пользователю возможность отменить произведенные обновления представлений, используя результаты последующих обновлений. Практическая реализация предложенного подхода однако не описана.

В современной работе [7] Бертосси и Салими рассматривают задачу обновления представлений применительно к подготовке данных для машинного обучения. Удаление кортежей из представления рассматривается как исключение зашумленных данных из обучающей выборки. Таким образом, авторы ограничиваются рассмотрением только одной операции удаления кортежа из представления.

При рассмотрении практической реализации обновления представлений необходимо рассмотреть стандарт SQL [30], в котором указываются ограничения на обновляемые представления. Основными ограничениями можно считать отсутствие в запросе на формирование представления ключевых слов DISTINCT, GROUP BY, HAVING, кроме того, атрибуты, перечисленные в разделе SELECT, могут появляться в этом списке не более одного раза. Все эти ограничения делают обязательным наличие однозначного соответствия строки и столбца с обновляемым значением со строкой и столбцом отношения БД.

В работах [56, 57] Мосунага с соавторами рассматривает задачу обновления представлений с использованием триггеров в СУБД PostgreSQL. Производится оценка вычислительной сложности алгоритмов обновления. Однако, общего решения не предлагается, кроме того, авторы сами указывают на то, что данное решение является неэффективным по быстродействию.

В соответствии с произведенным обзором работ по обновлению представлений, остается *актуальной* задача построения универсального аппарата,

который позволил бы обновлять многотабличные представления, для которых одной записи в представлении соответствует несколько записей в хранимых таблицах.

1.5 Выводы по главе 1

Использование нетипизированных зависимостей включения является следствием не решенных семантических проблем на этапе проектирования схемы БД. Далее изложены признаки таких проблем и способ их устранения. Предложенные формальные системы нетипизированных зависимостей используют перестановки атрибутов, что не согласуется с технологическими принципами проектирования и эксплуатации БД: на логическом уровне атрибуты идентифицируются своим именем, а не позицией в ряду других атрибутов. Кроме того, некорректная вставка значения $NULL$ в свойства зависимостей (определенное значение может быть подчиненным неопределенному значению) приводит к потере важных свойств зависимостей, в том числе, свойства транзитивности. С практической точки зрения такое решение не позволит избавиться от излишних ссылочных ограничений целостности, что скажется на скорости обработки данных.

Проблеме обновления представлений до сих пор уделяется большое внимание в научных работах. Но на практике требуется наличие однозначного соответствия строки и столбца с обновляемым значением со строкой и столбцом отношения БД. Такое ограничение препятствует созданию многотабличных пользовательских приложений. Всякий раз приходится тратить значительные усилия на программирование операций обновления данных, если они компонуются из нескольких отношений. Ссылочная целостность при этом является препятствием, которое надо преодолевать при программировании логики приложения.

Анализ смежных работ, проведенный в данной главе, говорит об *актуальности* исследования целостности данных касательно типизированных зависимостей включения и во время поддержки целостности при обновлении многотабличных представлений.

ГЛАВА 2. ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ ОТНОШЕНИЙ НА ОСНОВЕ ТЕОРИИ ЗАВИСИМОСТЕЙ ВКЛЮЧЕНИЯ

В данной главе описываются ссылочные ограничения целостности, основанные на типизированных зависимостях включения. Вводится система аксиом для типизированных зависимостей включения, доказывается ее полнота и непротиворечивость. Рассматривается минимальное множество зависимостей включения, после чего приводятся алгоритмы формирования неизбыточного набора типизированных зависимостей включения. Рассматривается правило преобразования зависимостей включения к типизированному виду. Описывается способ внедрения рассмотренной теории в существующие СУБД.

2.1 Теория типизированных зависимостей включения

В определении 1.1 представлена формулировка для типизированных зависимостей включения без учета неопределенных значений. Рассмотрим расширение этого понятия. Предварительно определим соответствующие друг другу кортежи при наличии неопределенных значений.

Определение 2.1. Кортеж $t_i[X]$ *соответствует* кортежу $t_j[X]$ по атрибутам X: $\left(t_j[X] \preccurlyeq t_i[X]\right)$, если для любого атрибута $A_l \in X$ выполнено одно из двух условий:

1) если $t_i[A_l] \neq NULL$, тогда $t_j[A_l] = t_i[A_l]$ или $t_j[A_l] = NULL$;
2) если $t_i[A_l] = NULL$, тогда $t_j[A_l] = NULL$.

Очевидно, что заданное в определении 2.1 отношение $t_j[X] \preccurlyeq t_i[X]$ является транзитивным. То есть, справедливо утверждение: если $t_j[X] \preccurlyeq t_i[X]$ и $t_i[X] \preccurlyeq t_m[X]$ тогда $t_j[X] \preccurlyeq t_m[X]$.

Определение 2.2. Зависимость включения $\sigma = R_j[X] \subsetneq R_i[X]$ от главного отношения $R_i[X]$ к подчиненному отношению $R_j[X]$ по атрибутам X *существует*, если для любого кортежа $t_j[X] \in R_j[X]$ имеется соответствующий кортеж $t_i[X]$ в отношении $R_i[X]$. Такую зависимость будем называть типизированной с допущением неопределенных значений.

Замечание. Кортеж $t_j[X] \in R_j[X]$ может иметь множество соответствующих кортежей в отношении $R_i[X]$. Для замены неопределенных значений в кортеже $t_j[X]$ могут быть выбраны только значения одноименных атрибутов одного из соответствующих кортежей отношения $R_i[X]$.

Обозначим множество зависимостей включения, определенных на схеме БД через Σ, а σ пусть будет произвольная зависимость, возможно, σ является элементом множества Σ.

Определение 2.3. Зависимость σ является логическим следствием множества зависимостей Σ ($\Sigma \vDash \sigma$), если данные в БД удовлетворяют всем зависимостям в Σ, тогда данные удовлетворяют зависимости σ. В этом случае зависимость σ будем называть *выполнимой*.

Заметим, что все зависимости Σ по определению 2.3 являются выполнимыми.

Представим систему аксиом, для зависимостей включения с возможными неопределенными значениями:

- **INN1** (рефлексивность): если $X \subseteq [R_i]$, тогда $R_i[X] \subsetneq R_i[X]$;
- **INN2** (проекция): если $R_j[Y] \subsetneq R_i[Y]$ и $X \subseteq Y$, тогда $R_j[X] \subsetneq R_i[X]$;
- **INN3** (транзитивность): если $R_j[X] \subsetneq R_i[X]$ и $R_i[X] \subsetneq R_l[X]$, тогда выполнено $R_j[X] \subsetneq R_l[X]$.

Отличие системы аксиом **INN1–INN3** от системы **IND1–IND3**, кроме допущения неопределенных значений, состоит в отсутствии перестановок в аксиоме **IND2**. В типизированных зависимостях включения могут быть сопоставлены друг другу только одноименные атрибуты, а на каком они находятся месте не важно. Это соответствует существующим технологиям БД: на логическом уровне атрибуты идентифицируются своим именем, а не своей позицией в наборе значений.

Заметим, что аксиомы **INN1–INN3** задают правила вывода.

Определение 2.4. Зависимость σ *выводима* из Σ за счет системы аксиом ($\Sigma \vdash \sigma$), если при применении аксиом к зависимостям Σ за конечное число шагов будет получена зависимость σ.

Для любой системы аксиом, прежде всего, необходимо показать ее надежность (непротиворечивость). Для этого покажем, что если зависимость σ выводима из множества Σ с использованием системы аксиом, то она является логическим следствием Σ: $\Sigma \vdash \sigma \Rightarrow \Sigma \vDash \sigma$.

Теорема 2.1. (Непротиворечивость) Система аксиом **INN1–INN3** непротиворечива.

Доказательство. Последовательно докажем непротиворечивость каждой из аксиом.

Рефлексия. Рассмотрим произвольный кортеж $t \in R_i$. В соответствии с определениями 2.1 и 2.2 кортеж t всегда будет соответствовать сам себе для любого $X \subseteq [R_i]$, что доказывает непротиворечивость аксиомы **INN1**.

Проекция. Предположим, что зависимость $R_j[X] \subsetneq R_i[X]$ не выполнена. Тогда допустимы реализации R_j и R_i такие, что существует кортеж $t_j \in R_j$, для которого нет соответствующего кортежа в отношении R_i по атрибутам X. Поскольку выполнена зависимость $R_j[Y] \subsetneq R_i[Y]$, для кортежа t_j

существует соответствующий кортеж $t_i \in R_i$ по атрибутам Y, то есть $t_j[Y] \preccurlyeq t_i[Y]$. Поскольку $X \subseteq Y$, то $t_j[X] \preccurlyeq t_i[X]$, что противоречит предположению об отсутствии соответствующего кортежа для t_j по атрибутам X. Это доказывает непротиворечивость аксиомы **INN2**.

Транзитивность. Рассмотрим произвольный кортеж $t_j \in R_j$. Поскольку выполнена зависимость $R_j[X] \subsetneq R_i[X]$, то существует кортеж $t_i \in R_i$: $t_j[X] \preccurlyeq t_i[X]$. Так как имеет место зависимость $R_i[X] \subsetneq R_l[X]$, то существует кортеж $t_l \in R_l$ такой, что $t_i[X] \preccurlyeq t_l[X]$. В силу транзитивности операции $\preccurlyeq$ выполнено условие $t_j[X] \preccurlyeq t_l[X]$, что доказывает непротиворечивость аксиомы **INN3**. *Теорема доказана*.

Замечание. Для отношений, удовлетворяющих условию аксиомы транзитивности, выполнены следующие соотношения:

$$R_j[X] \cap R_i[X] \subsetneq R_i[X] \cap R_l[X],$$

$$R_j[X] \cap R_i[X] \subsetneq R_j[X] \cap R_l[X],$$

$$R_j[X] \cap R_l[X] \subsetneq R_i[X] \cap R_l[X],$$

где $\cap$ – реляционный оператор пересечения.

В [118] после доказательства непротиворечивости системы аксиом функциональных зависимостей рассматриваются правила (теоремы), которые позволяют сократить вывод других правил и получать полезные свойства схемы БД. Наиболее полезными являются правила декомпозиции и объединения функциональных зависимостей. Аналогом правила декомпозиции является аксиома проекции **INN2**, а аналога правила объединения для зависимостей включения не существует. Из зависимости $R_j[X \cup Y] \subsetneq R_i[X \cup Y]$ выводимы зависимости $R_j[X] \subsetneq R_i[X]$ и $R_j[Y] \subsetneq R_i[Y]$. Обратное утверждение, к сожалению, не верно. Это является принципиальным отличием функциональных зависимостей от зависимостей включения. С одной стороны, это

упрощает доказательство полноты системы аксиом **INN1–INN3**. С другой стороны, отсутствует возможность удаления атрибутов в зависимостях при построении их минимального покрытия (рассмотрено далее), так как восстановить исходную зависимость уже не получится.

Рассмотрим новое доказательство полноты системы аксиом **INN1–INN3**, в котором учтены ранее высказанные замечания.

Теорема 2.2. (Полнота) Система аксиом **INN1–INN3** полна.

Доказательство. Необходимо показать, что если зависимость $\sigma = R_j[X] \subsetneq R_i[X]$ выполнима ($\Sigma \vDash \sigma$), то она выводима: $\Sigma \vdash \sigma$.

Для того, чтобы зависимость σ была выводима, достаточным условием является существование цепочки выполнимых зависимостей:

$$\begin{gathered} R_j[Y_1] \subsetneq R'_1[Y_1], \\ R'_1[Y_2] \subsetneq R'_2[Y_2], \\ \dots \\ R'_k[Y_{k+1}] \subsetneq R_i[Y_{k+1}], \end{gathered} \tag{2.1.1}$$

где $X \subseteq Y_l, l = 1, 2, \dots, k+1$. Действительно, по аксиомам проекции (**INN2**) и транзитивности (**INN3**), а в случае $i = j$ по аксиоме рефлексии (**INN1**), получаем выводимость зависимости σ.

Предположим, что зависимость σ не выводима. Тогда любая последовательность (2.1.1) содержит, по крайней мере, одну не выполнимую зависимость, пусть это будет $\sigma' = R'_{m-1}[Y_m] \subsetneq R_m[Y_m]$, где $Y_m = X$, и/или зависимость σ' выполнима, но $X \not\subseteq Y_m$. В этом случае в R'_{m-1}, а следовательно, в R_j может существовать кортеж t, которому нет соответствующего кортежа в отношении R'_{m-1}, а следовательно, в R_i по атрибутам X. Получили, что зависи-

мость σ не выполнима. Допустим, что в Σ есть зависимости, которые препятствуют появлению кортежа t в R'_{m-1} по атрибутам X. Поскольку это должно выполняться для любого состояния БД, удовлетворяющего Σ, то зависимость $R'_{m-1}[X] \subsetneq R_i[X]$ является выполнимой и ее можно использовать вместо не выполнимого участка последовательности (2.1.1). Полученные противоречия доказывают теорему.

Заметим, что предложенную схему доказательства достаточно просто распространить на нетипизированные зависимости включения. Однако, по выше названным причинам данный вид зависимостей в работе не рассматривается.

Аксиомы **INN1–INN3** задают правила вывода для зависимостей включения. Следовательно, они могут быть использованы для поиска выводимых (избыточных) зависимостей в Σ.

На основании рассмотренной теории зависимостей включения можно сформулировать общее правило преобразования однородных, но разноименных атрибутов. Как следствие, исчезнет необходимость использования нетипизированных зависимостей включения.

Правило. Производим сравнение однородных атрибутов. Для каждой совокупности однородных атрибутов находим название (A), которое их объединяет. Далее определяем различия семантики между этими атрибутами и присваиваем им имена (B_i). Старые атрибуты заменяем атрибутами A и B_i. Перестроение схемы начинаем с переопределения зависимостей.

2.2 Минимальное покрытие множества зависимостей включения

В данной диссертационной работе предлагается формальная теория для типизированных зависимостей включения при наличии неопределенных

значений. После описания теории зависимостей включения необходимо применить их на схеме БД. Важным аспектом для определения зависимостей включения на схеме БД является поиск минимально-достаточного множества зависимостей включения. Если будет определено недостаточное множество зависимостей включения, тогда ссылочная целостность данных не будет поддерживаться в такой БД. В случае, если будут установлены избыточные зависимости включения, это скажется на вычислительных затратах системы. Разработанная теория может быть использована для построения неизбыточного множества ссылочных ограничений целостности, которая описана в данном разделе.

На практике ссылочная целостность, теоретической основой которой являются зависимости включения, реализуется СУБД в виде первичных и внешних ключей. Эти ограничения необходимо хранить и модифицировать в процессе работы СУБД, что требует дополнительной памяти и времени. Поэтому целесообразно избавиться от избыточных зависимостей в множестве зависимостей включения Σ.

В предыдущем разделе доказано, что выводимая зависимость является выполнимой. Следовательно, ее можно удалить без всяких нежелательных последствий для БД: множество допустимых состояний БД останется без изменений. Поиск выводимых зависимостей напрямую является экспоненциальной задачей, поэтому воспользуемся известным аппаратом построения замыканий [118] для функциональных зависимостей. Аналогичный аппарат без доказательства корректности использован в работе [58]. Адаптируем эти результаты для типизированных зависимостей включения.

Определение 2.5. *Замыканием отношения* R_i на множестве зависимостей Σ относительно атрибутов X будем называть множество отношений $R_i^+[X]$, где $R_j \in R_i^+[X]$, если зависимость $\sigma = R_j[X] \subseteq R_i[X]$ выводима из Σ за счет аксиом **INN1–INN3**, то есть $\Sigma \vdash \sigma$.

Рассмотрим алгоритм построения замыкания. Текущее замыкание обозначим $R_i^*[X]$. Будем считать, что используемые в алгоритме множества имеют глобальные имена и их не надо передавать в процедуру через параметры.

Алг. 1. CLOSURE(OUT $R_i^*[X]$)

$R_i^*[X] = \emptyset$;
if $X_i^*[R_i] \neq \emptyset$
then
    exit Алг.
$R_i^*[X] = R_i$
$substitution$=**true**
while $substitution$
    $substitution$=**false**
    for each $\{R_l[Y] \subsetneq R_m[Y]\}$ **in** Σ
        if $R_m \in R_i^*[X]$ **and** $R_l \notin R_i^*[X]$ **and** $X \subseteq Y$
        then
            $R_i^*[X] = R_i^*[X] \cup R_l$
            $substitution$=**true**

В Алг. 1 внешний цикл WHILE не имеет явного ограничения. Однако, для выполнения следующего цикла необходимо дополнения хотя бы одного отношения к замыканию во внутреннем цикле FOR. Следовательно, максимальное количество итераций в алгоритме равно $n \times k$, где n – количество зависимостей в множестве Σ и k – количество отношений в БД.

Теорема 2.3. (Замыкание) Алг. 1 корректно формирует множество $R_i^+[X]$.

Доказательство. Пусть R_j произвольное отношение и X произвольное множество атрибутов. Необходимо показать, что $R_j \in R_i^+[X]$ тогда и только тогда, когда $R_j \in R_i^*[X]$, или $R_i^+[X] = R_i^*[X]$.

1. *Необходимость*. Пусть $R_j \in R_i^*[X]$. Множество $R_i^*[X]$ формируется в трех операторах:

а) $R_i^*[X] = \emptyset$. Множество $R_i^*[X]$ остается пустым, если X содержит атрибуты, которых нет в отношении R_i. По аксиомам **INN1–INN3** в этом случае также ничего не выводимо: $R_i^+[X] = \emptyset$.

б) $R_i^*[X] = R_i$. При выполнении условия $X \subseteq [R_i]$ замыкание $R_i^+[X]$ также будет содержать R_i по аксиоме рефлексии **INN1**.

в) $R_i^*[X] = R_i^*[X] \cup R_l$, если выполнены условия $R_l[Y] \subsetneq R_m[Y] \in \Sigma$, $R_m \in R_i^*[X]$, $R_l \notin R_i^*[X]$ и $X \subseteq Y$. По индукции покажем, что $R_l \in R_i^+[X]$. Базис индукции соответствует варианту (б). Предположим, что все отношения в $R_i^*[X]$ до появления зависимости $R_l[Y] \subsetneq R_m[Y]$ соответствуют выводимым зависимостям, то есть зависимость $R_m[Y] \subsetneq R_i[Y]$. Тогда по аксиоме проекции **INN2** выводима зависимость $R_l[X] \subsetneq R_m[X]$, так как $X \subseteq Y$, и по аксиоме транзитивности **INN3** имеем $R_l[X] \subsetneq R_i[X]$. Это верно для любого l, где $R_l \in R_i^*[X]$, в том числе для $l = j$. Следовательно, $R_i^*[X] \subseteq R_i^+[X]$.

2. *Достаточность*. Пусть $R_j \in R_i^+[X]$. Тогда существует k строк вывода, где последней строкой является зависимость $R_j[X] \subsetneq R_i[X]$. При $k = 1$ имеем $j = i$, либо $R_j[X] \subsetneq R_i[X] \in \Sigma$. В обоих случаях по алгоритму отношение R_j будет присоединено к $R_i^*[X]$. Пусть условие выполнимости имеет место для всех зависимостей, вывод которых содержит не более $k - 1$ строк. Кроме вариантов, рассмотренных для случая $k - 1$, зависимость $R_j[X] \subsetneq R_i[X]$ может быть получена за счет аксиомы проекции **INN2** из зависимости $R_j[Y] \subsetneq R_i[Y]$, где $X \subseteq Y$. Поскольку R_i уже содержится в $R_i^*[X]$, и если R_j еще нет в $R_i^*[X]$, то по алгоритму R_j будет присоединено к $R_i^*[X]$, поскольку выполнены все три условия оператора IF.

Кроме того, зависимость $R_j[X] \subsetneq R_i[X]$ может быть получена за счет аксиомы транзитивности **INN3** и R_j пока нет в $R_i^*[X]$. Тогда должно существовать отношение R_l, что зависимость $R_l[Y] \subsetneq R_i[Y]$, где $X \subseteq Y$, выводима, и зависимость $R_j[Z] \subsetneq R_l[Z]$, где $X \subseteq Z$, принадлежит Σ. По предположению $j \neq l$. Тогда $R_l \in R_i^*[X]$, так как цепочка вывода зависимости $R_l[Y] \subsetneq R_i[Y]$ короче, чем k. Поскольку вновь выполнены все условия оператора IF, то R_j будет принадлежать $R_i^*[X]$. Следовательно, $R_i^+[X] \subseteq R_i^*[X]$. *Теорема доказана*.

Имея в распоряжении полиномиальный алгоритм поиска избыточных зависимостей, осталось воспользоваться им для построения неизбыточного множества типизированных зависимостей включения. Такое множество в [118] называется минимальным покрытием.

Алг. 2. MIN-COVER(IN Σ; OUT Σ)

for each $\{R_j[X] \subsetneq R_i[X]\}$ **in** Σ
    if $R_j \in$ CLOSURE$(R_i^*[X])$
    then
        $\Sigma = \Sigma - \{R_j[X] \subsetneq R_i[X]\}$

С учетом количества итераций в Алг. 1 результирующее количество итераций в Алг. 2 будет равно n^2k.

После рассмотрения Алг. 2 закономерным является вопрос об эквивалентности зависимостей Σ на входе и на выходе алгоритма. Однако, уже доказанная связь между замыканием и выводимостью и то, что все выводимые зависимости являются выполнимыми, гарантирует одни те же ограничения на допустимые состояния БД со стороны зависимостей включения до и после работы Алг. 2

2.3 Автоматизация построения неизбыточного набора зависимостей включения

В данном разделе рассматривается вопрос автоматизации формирования ссылочных ограничений целостности, основанных на частном случае типизированных зависимостей включения. Такой подход становится эффективным при условии корректного построения схемы БД [113, 115, 118], в основе которого используются функциональные зависимости. В разделе рассматривается специальный вид ацикличности схемы БД, предложены алгоритмы поиска ссылочных ограничений целостности, основанные на неизбыточном множестве ациклических связей, доказана корректность результатов построений.

2.3.1 Формализация задачи

С использованием традиционных для БД обозначений формализуем постановку задачи. Путь $U = \{A_1, A_2, \dots, A_1\}$ – универсум (множество атрибутов в БД); $[R_i]$ – схема отношения R_i (множество атрибутов, на котором определено отношение R_i), $[R_i] \subseteq U$; $\mathfrak{R} = (R_1,\ R_2, \dots,\ R_k)$ – БД; $S = \{[R_1], [R_2], \dots, [R_k]\}$ – схема БД, $1 \leq i \leq k$. Для установления связей между отношениями используются следующие условия: каждый кортеж подчиненного (внешнего) отношения должен иметь соответствующий кортеж (кортежи) главного отношения. Причем значения одноименных атрибутов должны совпадать. Допускается наличие неопределенного значения атрибута в подчиненном отношении (если он не ключевой), которое соответствует определенному значению в главном отношении. Тип связи между отношениями соответствует количеству сопоставленных друг другу кортежей в связанных отношениях.

В данном разделе рассмотрим практическое использование типизированных зависимостей включения. Будем предполагать, что связи

устанавливаются по множеству одноименных атрибутов. При этом будем считать, что одноименные атрибуты являются однородными, то есть отражают одну и ту же характеристику. Как следствие, такие атрибуты должны быть однотипными. Такое дополнительное ограничение на связи полностью согласуется с общепринятым свойством связей на схеме БД: они отражают количественное соотнесение кортежей в отношениях друг с другом и не обладают какой-либо семантикой. Связывания различных по смыслу атрибутов, прежде всего, является следствием не решенных семантических проблем на этапе описательной части проекта БД. Как следствие, появляются проблемы при определении функциональных зависимостей для связываемых атрибутов. Так, в примере 1.1 [47] оставшаяся зависимость включения после подстановки правила преобразования и после удаления избыточной зависимости оказывается установленной для атрибутов-синонимов. Таким атрибутам соответствует тривиальная функциональная зависимость. То есть решение семантических проблем необходимо на подготовительном этапе проектирования схемы БД. Это позволяет избавиться от необходимости использования нетипизированных связей и зависимостей включения.

В прикладных задачах часто приходится иметь дело с неопределенными значениями атрибутов, когда некоторые характеристики объектов, за исключением идентификаторов (первичных ключей), не определены. В этом случае условие совпадения значений в проекциях $\pi_X(R_j) \subseteq \pi_X(R_i)$ (определение 1.1) может быть не выполнено. То есть в отношении R_j могут быть кортежи с неопределенными значениями. Этим кортежам соответствуют кортежи в главном отношении R_i, где соответствующие связям атрибуты принимают определенные значения. В предлагаемом подходе такое соответствие будем называть расширенной связью. Далее будет введено формальное определение связи в соответствии

с этим дополнением.

Предварительно рассмотрим пример, который служит демонстрацией ситуации, когда между отношениями должна быть установлена связь 1:1. Заметим, что в рамках классической теории проектирования БД такой тип связи не может появиться между отношениями, сформированными на основе минимального покрытия функциональных зависимостей.

Пусть A_1 – «Номер сотрудника», A_2 – «Фамилия сотрудника», A_3– «Дата увольнения сотрудника». Существуют функциональные зависимости: $A_1 \to A_2$ и $A_1 \to A_3$. Области определения этих зависимостей различаются: первая зависимость определена для всей прикладной области (всего множества сотрудников). Вторая зависимость определена только для уволенных сотрудников. Это является основанием для построения декомпозиции: $R_i[A_1, A_2]$ и $R_j[A_1, A_3]$, а между отношениями R_i и R_j устанавливается связь **1:1**, где R_i – главное отношение, R_j – подчиненное (внешнее) отношение. Применение такой декомпозиции позволяет получить решение проблемы неопределенных значений. Так как в рамках третьей нормальной формы будет получено отношение $R_l[A_1, A_2, A_3]$, в котором атрибут A_3 принимает неопределенное значение для всех не уволенных на данный момент сотрудников. В данной работе рассмотренный пример демонстрирует основание для появления типа связи **1:1**. Понятно, что формальное решение этой проблемы должно основываться на областях определения функциональной зависимости. Однако это выходит за рамки данной работы.

Пусть $PK(R_i)$, либо $PK(i)$, первичный ключ отношения R_i. Обозначим $L(i, j, X)$ – связь типа **1:1,** либо типа **1:M** от главного отношения R_i к подчиненному (внешнему) отношению R_j, где X множество атрибутов, по которым установлена связь. Пусть $L_1(i, j, X)$ – связь типа **1:1**, установленная от отношения R_i к отношению R_j; $L_M(i, j, X)$ – связь типа **1:M** от отношения

R_i к отношению R_j. Заметим, что произвольное отношение R_i может содержать несколько первичных ключей, и с ним может быть установлено несколько связей, в которых R_i будет являться главным или подчиненным отношением.

Далее рассмотрим определения, в которых формулируется возможность установления связей между отношениями БД, которые могут быть использованы в качестве расширенных ссылочных ограничений целостности. В работе [105] была сделана попытка полностью автоматизировать процесс построения связей на схеме БД. Однако впоследствии было получено условие, ограничивающее область применения такого подхода. Принципиальное отличие в том, что окончательное решение об установлении ссылочных ограничений целостности может сделать только проектировщик БД. Однако, следующие определения помогут автоматизировать процесс поиска ссылочных ограничений целостности.

Определение 2.6. *Связь* $L_1(i, j, X)$ может быть установлена между отношениями R_i и R_j, если они имеют однородные первичные ключи $X = PK(R_i) = PK(R_j)$ и для любых реализаций отношений R_i и R_j выполнено $R_j[X] \subseteq R_i[\mathrm{X}]$.

В определении 2.6 использована более короткая запись $R_j[X] \subseteq R_i[\mathrm{X}]$ для выражения $\pi_X(R_j) \subseteq \pi_X(R_i)$. Данное определение не допускает наличие неопределенных значений для атрибутов X, поскольку они в обоих отношениях являются компонентами первичного ключа.

Определение 2.7. *Связь* $L_M(i, j, X)$ может быть установлена между отношениями R_i и R_j, если $PK(R_i) \neq PK(R_j)$ и $PK(R_i) \subseteq [R_j]$.

В определении 2.7 отношение R_j может содержать кортежи с неопределенными значениями атрибутов, которые не являются компонентами первичного ключа. Заметим, что определения 2.6 и 2.7

соответствуют типизированным зависимостям включения, если соответствующим связям будет установлено свойство ссылочной целостности данных. Далее это свойство поддерживается СУБД за счет создания внешних ключей. Свойство ссылочного ограничения целостности для связи $L_M(i, j, X)$ не гарантирует выполнения соотношения $R_j[X] \subseteq R_i[\mathrm{X}]$, где $X = [R_i] \cap [R_j]$, так как атрибуты, не принадлежащие первичному ключу отношения, могут принимать неопределенные значения. Однако, в главном отношении тогда им должны соответствовать определенные значения. Ссылочная целостность реализуется следующим образом: неопределенное значение атрибута в подчиненном отношении может быть заменено только тем определенным значением из соответствующего кортежа в R_i, в котором остальные определенные значения совпадают. Поиск связей, соответствующих определениям 2.6 и 2.7, достаточно просто алгоритмизируется, что позволяет выявлять большинство ссылочных ограничений целостности в автоматическом режиме.

В технологиях БД [22], выделены дополнительные типы связей: 1) «один или несколько обязательных», 2) «один и только один», 3) «ноль или один необязательный», 4) «ноль или многие необязательные». В настоящее время рассматриваются только парные связи на схеме БД, то есть связь 1 является модификацией (усилением) связи **1:M** со стороны подчиненного отношения, связь 2 – является модификацией (усилением) связи **1:1** со стороны подчиненного отношения, связь 3 аналогична связи **1:1** для подчиненного отношения, связь 4 аналогична связи **1:M** со стороны подчиненного отношения. Другие типы связей, соответствующие типу связи 3 со стороны главного отношения, не попадают под определение зависимости включения и поэтому не могут являться ссылочным ограничением целостности. Такие типы связей далее не будут учитываться. Исключение представляет тип связи **1:M**, который предполагает наличие

неопределенных значений одноименных атрибутов в подчиненном отношении при условии, что они не ключевые. Такой вариант связи соответствует введенной ранее расширенной связи.

Таким образом, автоматизация формирования ссылочных ограничений целостности должна соответствовать расширенной связи и основываться на определениях 2.1 и 2.2.

2.3.2 Свойство ацикличности схем баз данных

В работе [58] свойство ацикличности зависимостей включения произвольного вида формулируется следующим образом: множество зависимостей является ациклическим, если не существует зависимости $R[X] \subseteq R[Y]$, $X \neq Y$ и не существует последовательности отношений $R_1, R_2, \ldots, R_n$ $(n > 1)$, для которых выполнено $R_1[X_1] \subseteq R_2[Y_2]$, $R_2[X_2] \subseteq R_3[Y_3]$,…, $R_n[X_n] \subseteq R_1[Y_1]$. В этом случае предлагается полиномиальный алгоритм проверки ацикличности зависимостей включения, основанный на построении замыкания множества отношений. Рассмотрим простейший пример с типизированными зависимостями включения $R_1[X_1] \subseteq R_2[X_1]$, $R_2[X_1] \subseteq R_3[X_1]$, и пусть атрибуты X_1 в отношении R_2 допускают неопределенные значения. Тогда замыкание $R_1[X_1] \subseteq R_3[X_1]$ не выполнено, поскольку не выполнена зависимость $R_2[X_1] \subseteq R_3[X_1]$. Это говорит о том, что учет расширенных связей не позволяет использовать аппарат выводимости и построение замыкания. В предлагаемом подходе условие $R_2[X_1] \subseteq R_3[X_1]$ выполнено, если X_1 является первичным ключом в R_2, а в первичном ключе неопределенные значения не допускаются. Это несколько сужает область применимости, но позволяет формировать корректные построения с учетом расширенных связей.

Пусть задана совокупность отношений БД $\mathfrak{R} = (R_1, R_2, \ldots, R_k)$. Условие ацикличности для подмножества отношений сформулируем с

учетом расширенных связей.

Определение 2.8. Множество всех отношений БД $\mathfrak{R} = (R_1, R_2, \dots, R_k)$ будем называть *ациклическим*, если не существует такого подмножества:

$$\{R_{m(1)}, R_{m(2)}, \dots, R_{m(s)}\} \subseteq \mathfrak{R}, \quad (2.3.1)$$

для которого установлены связи:

$$L(m(1), m(2), X_1), L(m(2), m(3), X_2), \dots, L(m(s), m(1), X_s) \quad (2.3.2)$$

s > 1 и $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_s$. Иначе множество отношений $\mathfrak{R}$ будем называть *циклическим*.

В последовательности (2.3.2) допускается наличие расширенных связей.

Определение 2.9. *Гиперграфом* на схеме БД называется пара $G = (U, S)$ [115], вершинами U являются атрибуты БД, гиперребрами S являются схемы отношений $[R_i]$, в которые объединены вершины – атрибуты, входящие на схему отношения $[R_i]$. Гиперграф является *циклическим*, если для произвольного атрибута A существует последовательность $[R'_1], [R'_2], \dots, [R'_m]$ и выполнено: $A \in [R'_1]$ и $A \in [R'_m]$, и $[R'_i] \cap [R'_{i+1}] \neq \emptyset$ для $1 \leq i \leq m - 1$.

В статье [6] рассмотрены различные эквивалентные условия ацикличности $\mathfrak{R}$. Наиболее важное условие, анализ которого приведен в работе [115], устанавливает, что схема БД является ациклической, если отсутствуют циклы в ассоциированном гиперграфе. Из предположения об отсутствии вложенных схем отношений: $[R_i] \subseteq [R_j]$, $i \neq j$, следует, что ассоциированный гиперграф редуцирован.

Поскольку ребра в гиперграфе соответствуют отношениям, а узлы – атрибутам, то «связанными» оказываются отношения с одноименными

атрибутами. Такой тип связи соответствует типизированным зависимостям включения. Для нетипизированных зависимостей такая интерпретация отсутствует.

По аналогии с [105] имеет место терема.

Теорема 2.4. (Цикличность). Если для схемы $\mathfrak{R}$ выполнены условия цикличности определения 2.8, то соответствующий $\mathfrak{R}$ гиперграф будет циклическим.

Доказательство. Рассмотрим произвольный набор ребер гиперграфа $\{R_{m(1)}, R_{m(2)}, \ldots, R_{m(s)}\} \subseteq \mathfrak{R}$. Допустим, что существует последовательность $L(m(1), m(2), X_1), L(m(2), m(3), X_2), \ldots, L(m(s), m(1), X_s)$, следовательно, $R_{m(i)} \cap R_{m(i+1)} \neq \emptyset$, $i = 1, 2, \ldots, s$, где $m(s+1) = m(1)$. Ребра $\{R_{m(1)}, R_{m(2)}, \ldots, R_{m(s)}\} \subseteq \mathfrak{R}$ образуют цикл в ассоциированном гиперграфе. Следовательно, он будет циклическим.

Для того, чтобы показать, что обратное утверждение не верно достаточно построить один контрпример. Для этого в качестве основы используем пример 13.15 из работы [115]. Пусть БД $\mathfrak{R} = \{R_1, R_2, R_3\}$ имеет следующие схемы отношений $R_1[ABC]$, $R_2[BD]$, $R_3[CD]$. Допустим, что для формирования таких схем использовались функциональные зависимости $\{BC \to A, B \to D, C \to D\}$. Гиперграф для данной совокупности отношений является циклическим. А по определению 2.8 схема такой БД не является циклической. Существующие связи $L_M(2,1,B)$ и $L_M(3,1,C)$ подтверждают это. Следовательно, требование ацикличности по определению 2.8 является более слабым, чем требование в [6] и большее количество схем БД оказываются ациклическими. *Теорема доказана.*

2.3.3 Удаление избыточных связей

Допустим, что БД $\mathfrak{R}$ является ациклической в соответствии с определением 2.8. Связь $L(i, j, X)$ ограничивает допустимые кортежи T_i в

отношении R_j.

Определение 2.10. Связь $L(i,j,X)$ является *избыточной*, если существует последовательность $L(l,j,Y_l)$, $l = 1,\dots,m$, где $\bigcap_{l=1}^{m} T_l \subseteq T_i$.

Заметим, что определение избыточной связи не является структурным (поиск «простых» путей на схеме БД [114]), а опирается на содержимое кортежей БД при реализации ссылочной целостности БД. В определении 2.10 не учитываются избыточные связи, в основе которых лежат нетипизированные зависимости включения [9, 45, 44]. Кроме того, под определение не попадают связи, атрибуты которых не являются первичным ключом для главного отношения. Однако, такой тип связи поддерживается практически всеми существующими СУБД, решение семантических проблем на этапе проектирования СУБД позволяет избавиться от других типов связи. Перечисленные аргументы еще раз доказывают практическую значимость данного раздела. Далее рассмотрим последовательность проектирования БД с детальной проработкой использования рассматриваемых типов связи.

При классическом подходе [118] к проектированию БД, после решения семантических проблем с семантикой атрибутов (название атрибута должно определять множество значений, которые атрибут может принимать), строится минимальное покрытие множества функциональных зависимостей с учетом поправок, сделанных в работе [119]. На основе минимального покрытия строятся отношения БД $\mathfrak{R} = \{R_1, R_2, \dots, R_k\}$, гарантированно удовлетворяющие требованиям третьей нормальной формы. Определения 2.6 и 2.7 позволяют автоматизировать формирование связей $L_M(i,j,X)$ и $L_1(i,j,X)$ с последующим удалением избыточных связей.

Связи $L_M(i,j,X)$ не могут быть автоматически установлены за счет простого перебора в соответствии с определением 2.7. Такая связь предполагает наличие ссылочного ограничения целостности и определяется семантикой прикладной области. На схеме БД «ВУЗ», взятой из Рис. 1,

должны быть установлены следующие связи:

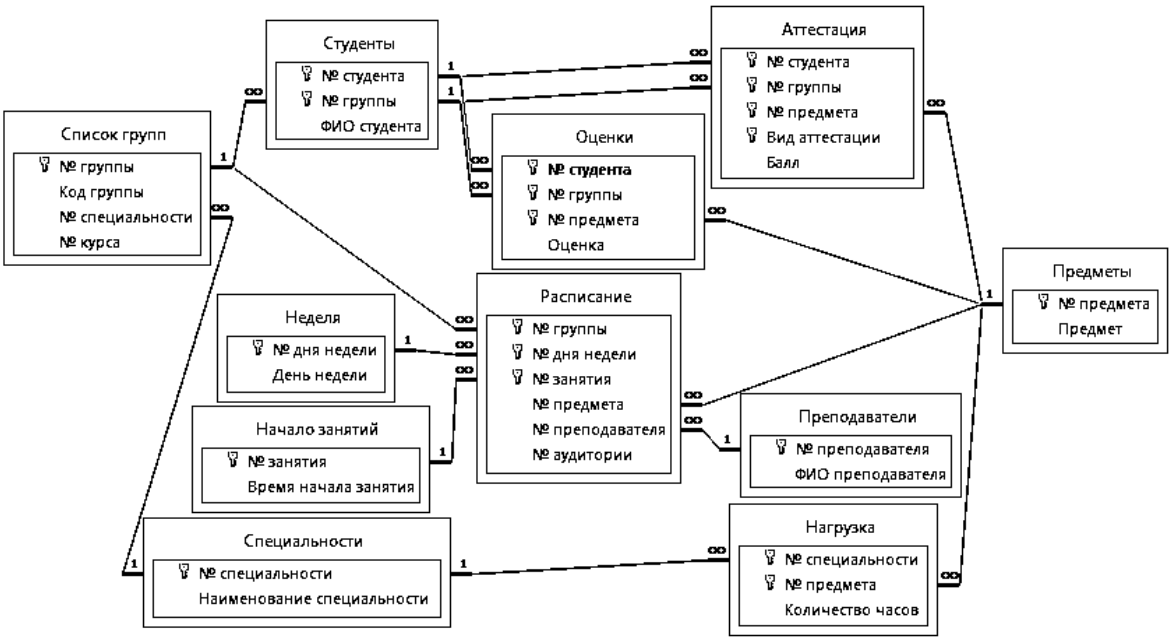

Рис. 1. Полная схема БД ВУЗ

$L_M(2,1,<$ № группы $>)$, $L_M(2,4,<$ № группы $>)$, $L_M(2,5,<$ № группы $>)$, $L_M(1,4,<$ № студента, № группы $>)$, $L_M(1,5,<$ № студента, № группы $>)$, $L_M(3,4,<$ № предмета $>)$, $L_M(3,5,<$ № предмета $>)$. Формально, по определению 2.7 должна быть установлена связь $L_M(4,5,<$ № студента, № группы, № предмета $>)$. Последняя связь предполагает следующее ограничение целостности: аттестация студентов может осуществляться только по тем предметам, по которым проставлены оценки. Поскольку это противоречит прикладной области, то связь не должна присутствовать на схеме БД. Проектировщику БД в этом случае необходимо ответить на вопрос: нужно или нет данное ограничение целостности.

При дополнении связи $L_1(i,j,X)$ предварительно определяется роль каждого отношения: какое из пары отношений является главным, а какое – внешним. На этапе проектирования данные обычно отсутствуют, поэтому условие $\pi_V(R_j) \subseteq \pi_V(R_i)$, определенное на первичном ключе V отношений

R_i и R_j, не может быть установлено с использованием каких-либо алгоритмов. В данном случае проектировщик должен опираться на семантику прикладной области. При определении типа связи между R_i и R_j необходимо соотнести значения одноименных атрибутов $R_i \cap R_j$ в отношениях R_i и R_j. Возможен один из трех альтернативных вариантов:

1. Множество векторов значений атрибутов V в отношении R_i является подмножеством векторов значений в R_j.
2. Множество векторов значений атрибутов V в отношении R_j является подмножеством векторов значений в R_i.
3. Условия 1-го и 2-го вариантов могут быть нарушены.

В первом случае устанавливается связь $L_1(i, j, V)$, при выборе второго варианта устанавливается связь $L_1(j, i, V)$, в третьем варианте связь не устанавливается. Допустим, что этот выбор реализуется функцией $Ch(R_i, R_j, V)$, имеющей значения $L(i, j, V)$, либо пусто $(\emptyset)$.

Рассмотрим алгоритм для автоматизации формирования связей на схеме БД. Пусть L – текущее множество связей, k – количество отношений в текущей БД. Вычислительная сложность Алг. 3 является полиномиальной и оценивается формулой: $O(k^2)$.

Алг. 3. FORM_REL(OUT L)

```
L = ∅
for i = 1 TO k
    for each PK(R_i) in R_i
        for j = 1 TO k
            if i ≠ j
            then
                if PK(R_i) ∈ R_j
                then
                    L = L ∪ Ch(R_i, R_j, V)
```

В работе [58] сформулировано условие проверки на избыточность зависимости включения, однако, доказательство корректности этого условия не приводится. При доказательстве избыточности зависимости обычно используется выводимость [12]. Условие для определения избыточных связей, обычных и расширенных, докажем с использованием реализаций отношений (выполнимость), поскольку выводимость на связях не определялась.

Теорема 2.5. Связь $L(i,j,X)$ на схеме БД избыточна, если существует последовательность:

$$L(i,m(1),X_0), L(m(1),m(2),X_1), \dots, L\big(m(p),j,X_p\big) \tag{2.3.3}$$

и

$$X \subseteq PK(i) \subseteq X_s \subseteq R_{m(s)},\ s = 2,3,\dots,p, \tag{2.3.4}$$

где $m(i)$ – номера отношений.

Доказательство. Выполнение условий $PK(i) \subseteq R_j$ и $PK(i) \subseteq R_{m(1)}$ гарантировано при наличии связей $L(i,j,X)$ и $L(i,m(1),X_0)$ соответственно.

1) Предположим, что выполнены условия (2.3.3) и (2.3.4). Множество атрибутов первичного ключа $PK(i)$ отношения R_i присутствует в каждом из отношений:

$$R_i, R_{m(1)}, R_{m(2)}, \dots, R_{m(p)}, R_j \tag{2.3.5}$$

тогда по определению оно должно участвовать в связях (2.3.3), следовательно, $\pi_{PK(i)}(R_i) \supseteq \pi_{PK(i)}(R_{m(1)}) \supseteq \cdots \supseteq \pi_{PK(i)}(R_{m(p)}) \supseteq \pi_{PK(i)}(R_j)$. Из полученного соотношения для включений следует, что в R_j не может присутствовать кортеж со значением $PK(i)$, отличным от значений в других отношениях последовательности (2.3.5). Это условие сильнее ограничения связи $L(i,j,X)$ и, следовательно, эта связь избыточна по определению 2.10. Допустим, что связь $L(m(p),j,X_p)$ в (2.3.3) является расширенной, а связь $L(i,j,X)$ – нет. Остальные связи в (2.3.3) не могут быть расширенными, так как одноименные атрибуты в подчиненных отношениях являются ключевыми. Уже показано, что в отношении R_j не может быть кортеж с неопределенными значениями $PK(i)$. Следовательно, связь $L(i,j,X)$ избыточна.

2) Предположим, что последовательность (2.3.5) содержит отношение $R_{m(s)}$, в котором нет некоторых атрибутов $PK(i)$. Тогда после отношений $R_{m(s-1)}$ и $R_{m(s)}$ ограничений на атрибуты $PK(i) - [R_{m(s)}]$ нет. Эти атрибуты могут принимать произвольные значения, в том числе, отличные от значений в проекции $\pi_{PK(i)}(R_i)$. Следовательно, ограничения в цепочке (2.3.3) не достаточны для удаления связи $L(i,j,X)$.

3) Пусть имеет место условие (2.3.4), но отсутствует возможность построения (2.3.3), тогда атрибуты $PK(i)$ в R_j могут принимать произвольные значения. *Теорема доказана*.

Следствие. Условие (2.3.3) при $p = 1$ является достаточным для избыточности связи $L(i,j,X)$.

Замечание. Покажем, что избыточными могут быть только одиночные связи. Рассмотрим две последовательные (2.3.3) связи $L(i,v,X_i)$, $L(v,j,X_v)$, $v \neq m(s)$, $s = 1,2,\dots,p$. Удаление любой из $L(i,v,X_i)$ и $L(v,j,X_v)$ означает удаление ограничений связи $L(i,v,X_i)$ на R_v.

Использование условий (2.3.3) и (2.3.4) для поиска избыточной связи является экспоненциальной задачей. Так как связи соответствуют типизированным зависимостям включения, то к поиску избыточных связей применим Алг. 2 построения замыкания. Однако, анализировать алгоритм должен не зависимости, а связи, установленные с помощью первичных ключей отношений. На вход алгоритма подается сформированное множество связей L, на выходе то же самое множество, но только без избыточных связей.

Определим вычислительную сложность алгоритма. Пусть $|L|$ – количество связей в L до обработки его алгоритмом. На каждой итерации цикла DO WHILE к p должен присоединяться как минимум номер одного

Алг. 4. MIN-REL (IN L; OUT L)

```
for each L(l, j, X) in L
    q = 1
    p(q) = l
    changes =true
    do while changes
        for each L(v, w, X_q) in L where L(l, j, X) ≠ L(v, w, X_q)
            del_rel=false
            if v ∈ p[1, … , q] and PK(R_l) ⊆ R_w
            then
                if w = j
                then;
                    L = L – L(l, j, X)
                    exit do
                else
                    if w ∉ p[1, … , q]
                    then
                        q = q + 1
                        p(q) = w
                        del_rel = true
        if not del_rel
        then changes = false;
```

отношения, тогда Алг. 4 будет иметь следующую вычислительную сложность: $O(|L|^2k^2)$, где k – количество отношений БД. Степень при k обусловлена необходимостью проверок $v \in p[1,\dots,q]$ и $w \notin m[1,\dots,q]$. Корректность Алг. 3 доказывается по индукции (аналог теоремы 5.2 в [118]).

Допустим, для множества отношений $R_{p(1)}, R_{p(2)}, \dots, R_{p(m)}$ имеются связи $L_1(p(1), p(2), X_1)$, …, $L_1(p(m-1), p(m), X_{m-1})$. Допустим, что таких последовательностей несколько, тогда они могут иметь общие отношения. Причем, по определению 2.7 наличие связи $L_M(i, j, X)$ у отношения R_i гарантируют, что связи $L_M(m(l), j, X_l)$, $l = 1, \dots p$, также могут быть установлены. При удалении избыточных связей останутся связи $L_M\big(m(p), j, X_p\big)$, исходящие из последних отношений множества. Такие связи задают более жесткие ограничения, чем промежуточное отношение R_i. Следовательно, необходимо вернуть связи к отношениям, соответствующим реальным ограничениям. Количество связей определяется свойством, которое является следствием теоремы 2.5.

Свойство. Пусть связи $L_1(m(1), m(2), X_1)$, …, $L_1\big(m(p-1), m(p), X_{p-1}\big)$ установлены для отношений $R_{m(1)}, R_{m(2)}, \dots, R_{m(p)}$, пусть R_j произвольное отношение, тогда существует не более одной связи $L_M(m(l), j, X_l)$.

На схеме БД эти связи могут задавать дизъюнкцию ограничений на $PK(R_i)$, являющихся компонентами R_j. Однако существующие СУБД пока не могут поддерживать такой вид ограничений.

2.4 Внедрение в СУБД типизированных зависимостей включения

В разделе 2.1 был описан теоретический базис типизированных зависимостей включения при использовании неопределенных значений. Фактически, зависимости включения представляют собой бизнес-правила, которые задают новые ссылочные ограничения целостности. В данном разделе рассматриваются примеры бизнес-правил, после чего приводится способ их внедрения в качестве декларативного ограничения целостности в СУБД.

2.4.1 Применение зависимостей включения для реализации сложных бизнес-правила

Рассмотрим общий случай бизнес правила, основанного на двух отношениях R_1 и R_2: $R_1[A, B], R_2[A, C]$. Пусть установлена типизированная зависимость включения $R_2[A] \subseteq R_1[A]$ от главного отношения R_1 к подчиненному отношению R_2. В отношении R_1 могут находиться кортежи $t_1^{R_1} = (a_1, b_1)$ и $t_2^{R_1} = (a_1, b_2)$. В соответствии с ссылочной целостностью в подчиненном отношении R_2 могут находиться кортежи $t_1^{R_2} = (a_1, c_1)$. Наличие атрибута $t_1^{R_2} = (a_2, b_1)$ ограничивается типизированной зависимостью включения по причине отсутствия соответствующего значения a_2 в таблице R_1.

При установлении внешнего ключа от ссылающейся таблицы R_2 к главной таблице R_1 необходимо установить первичный ключ (PRIMARY KEY), либо ограничение уникальности (UNIQUE) на значения атрибута A в таблице R_1. В этом случае невозможно будет добавить в таблицу два атрибута $t_1^{R_1} = (a_1, b_1)$ и $t_2^{R_1} = (a_1, b_2)$, так как наличие в обоих кортежах значе-

ния a_1 нарушает условие уникальности. Ограничения на ссылающуюся таблицу R_2 останутся прежними и в нее нельзя будет добавить значение a_2 для атрибута A.

Применим новое бизнес-правило к фрагменту БД «ВУЗ» (полное описание которой представлено на Рис. 1), состоящей из двух таблиц «Студенты» и «Оценки». Для иллюстрации правила опишем каждое отношение:

R_1 = *Студенты* (№ студента, № группы, ФИО студента);

R_4 = *Оценки* (№ студента, № группы, № предмета, Оценка).

Подчеркнутыми атрибутами выделим ключевые атрибуты.

Пусть задано правило: студенты, которые не прошли контрольную точку не допускаются до экзамена. При установке данного ограничения необходимо добавить еще одну таблицу *Контрольные точки:*

R_6 = *Контольные точки* (№ Контрольной точки, № студента, № группы, Баллы);

Если в таблице «Контрольные точки» установить ограничение уникальности на пару отношений № студента, № группы, тогда можно будет установить внешний ключ от ссылающейся таблицы «Оценки» к главной таблице *Контрольные точки* по атрибутам № студента, № группы. Схема данных, состоящая из трех таблиц представлена на Рис. 2. Ограничение, задаваемое бизнес-правилом, на данном примере будет работать, однако, ограничение уникальности запретит наличие дублирующей записи с повторяющимся студентом. С прикладной точки зрения это говорит о том, что один студент может пройти только одну контрольную точку. Для того, чтобы для одного студента хранить информацию о всех контрольных точках, предлагается использовать модернизированную теорию типизированных зависимостей включения, которая позволяет наличие дублирующих значений в глав-

ном отношении по атрибутам, на которые ссылаются соответствующие атрибуты в подчиненном отношении. Для реализации рассматриваемого бизнес-правила необходимо внешний ключ от таблицы «Оценки» к таблице «Контрольные_точки» заменить на типизированную зависимость включения.

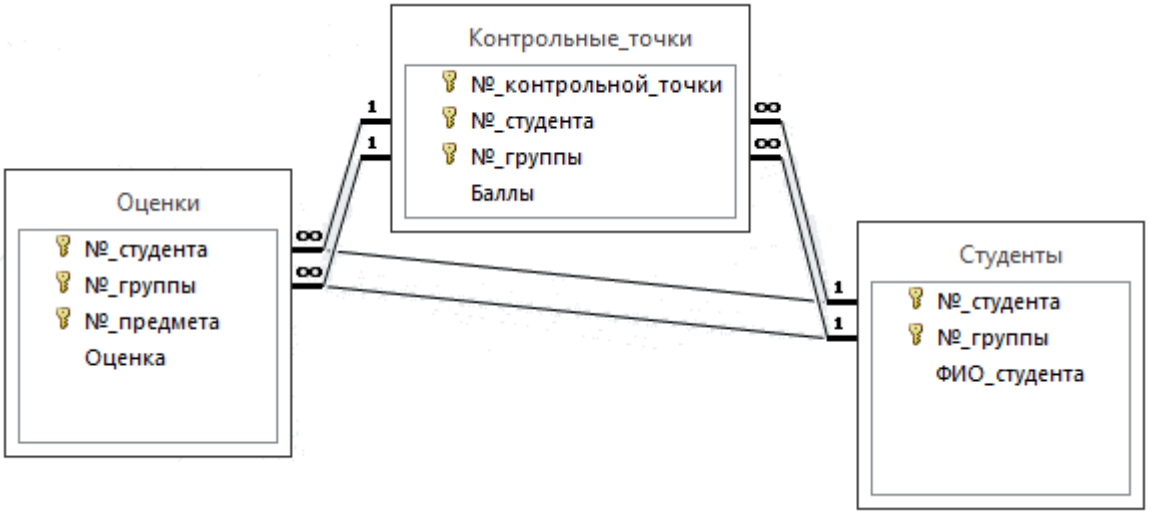

Рис. 2. Схема БД «Студенты-Оценки».

2.4.2 Реализация внедрения декларативного ограничения целостности в СУБД

Далее рассмотрим способы внедрения разрабатываемых в данной диссертационной работе бизнес-правил в свободную СУБД PostgreSQL путем создания собственных декларативных ограничений целостности для СУБД. Для этого была произведена следующая последовательность операций над СУБД:

1) Модифицирован язык описания данных (DDL) конкретной СУБД;
2) Модифицирован парсер СУБД;
3) Модифицирован словарь БД (системные таблицы);
4) Модифицирована подсистема контроля целостности данных;
5) Реализована политика каскадного обновления и удаления.

При модернизации языка СУБД необходимо было добавить в него новую конструкцию ограничения (DEPENDS ON), которую можно описать в

терминах формальной системы Бэкуса–Наура, которая представлена в Лист. 1. Введем новую форму задания ограничений:

Лист. 1: Установка ограничений целостности. Форма Бэкуса–Наура

```
CONSTRAINT constraint_name{

 { UNIQUE | PRIMARY KEY } "("column [, column] ...")"
 | FOREIGN KEY "("column [, column] ...")" REFERENCES [ schema .] table
"("column [, column] ...")" [ON DELETE {CASCADE | RESTRICT}]
 | DEPENDS "("column [, column] ...")" ON table   "("column [, column] ...")"
[ON DELETE {CASCADE | RESTRICT}]
 | CHECK "("condition")"}
```

Далее опишем способ установления рассмотренного бизнес-правила в таблицу «Оценки». В Лист. 2 предложен SQL запрос на создание таблицы с определением ограничения типизированной зависимости включения.

Лист. 2: Пример установления нового ограничения целостности

```
CREATE TABLE Оценки
(№_студента int,
№_группы int,
№_предмета int,
Оценка tinyint,

PRIMARY KEY (№_студента, №_группы, №_предмета),

CONSTRAINT fk_Оценки_Студенты FOREIGN KEY (№_студента,
№_группы) ON Студенты(№_студента, №_группы),

CONSTRAINT Dpnds_Оценки_Контрольные_точки DEPENDS
(№_студента, №_группы) ON Контрольные_точки(№_студента,
№_группы),
);
```

После успешного внедрения нового типа ссылочного ограничения целостности, основанного на типизированных зависимостях включения, СУБД должна самостоятельно контролировать выполнение всех ограничений целостности. Другими словами, можно рассмотреть новую последовательность проверок целостности для СУБД:

1) NOT NULL;
2) UNIQUE;
3) PRIMARY KEY;
4) FOREIGN KEY;
5) DEPENDS ON;
6) CHECK;

Новое ограничение целостности логично будет проверять сразу после ограничений внешнего ключа, где рассматриваются связи между таблицами.

2.4.3 Политика каскадного обновления и удаления

Политика каскадного обновления и удаления должна быть реализована в новом ограничении целостности по аналогии с существующей технологией каскадных обновлений, связанных с внешними ключами.

Рассмотрим политику каскадного обновления (удаления) для новой зависимости включения DEPENDS ON. Пусть производится обновление записи в «Контрольные_точки» (изменили №_студента), запускается каскадное обновление по связи «Оценки» → «Контрольные_точки», в таблице «Оценки» необходимо обновить соответствующую запись (соответствующий №_студента). В таблице «Оценки» соответствующую запись можно обновлять любым из заданных способов:

– NO ACTION;
– SET NULL;
– SET DEFAULT;

Рассмотрим пример каскадного удаления. Пусть пользователь удалил запись из главной таблицы «Контрольные_точки» (удалил определенный №_студента), тогда в соответствии с политикой каскадного удаления во внешней таблице «Оценки» должны быть произведены одна из операций SET NULL, SET DEFAULT, либо любая другая, поддерживаемая конкретной СУБД.

2.5 Выводы по главе 2

Отметим, что при рассмотрении формальной теории зависимостей включения в работе не потребовалось ограничиваться ациклическими зависимостями и ограничивать арность зависимостей, что делает предложенный аппарат достаточно универсальным. На практике циклические зависимости могут существенно снизить функциональные возможности БД, например, при дополнении новой информации. Однако эта проблема возникает на этапе проектирования БД, когда формируется множество Σ, и эта проблема решается на семантическом уровне. В качестве технологической поддержки предложенный аппарат должен быть расширен правилами и алгоритмами поиска циклических зависимостей.

Автоматизация поиска ссылочных ограничений целостности регламентируется определениями 2.6 и 2.7, что достаточно для большинства корректно сформированных схем БД. Анализ приложений показал, что использование нетипизированных зависимостей включения обусловлено нерешенными проблемами на этапе проектирования БД. В разделе 2.4 рассмотрены примеры использования нетипизированных зависимостей с анализом ошибок и предложено правило преобразования нетипизированной зависимости к типизированному виду, которое поможет решить семантические проблемы.

Основные результаты, описанные в данной главе, опубликованы в работах: [80, 81, 95, 96, 102, 103].

ГЛАВА 3. ЦЕЛОСТНОСТЬ МНОГОТАБЛИЧНЫХ ПРЕДСТАВЛЕНИЙ НА ОСНОВЕ КОММУТАТИВНЫХ ПРЕОБРАЗОВАНИЙ ДАННЫХ

В данной главе описываются теоретические аспекты обновления многотабличных представлений. Приводится метод коммутативных преобразований. Предлагаются аналитические формулы в терминах реляционной алгебры, осуществляющие операции обновления представлений. Формулируются и доказываются теоремы о корректности предложенных формул. Приводится архитектура для сопроцессора, который реализует операции обновления представлений.

Для демонстрации предлагаемых решений рассмотрим фрагмент схемы БД учебного заведения, где подчеркнуты ключевые атрибуты отношений:

Пример 1. БД «Вуз»

R_1 = *Студенты* (№ студента, № группы, ФИО студента);

R_2 = *Список групп* (№ группы, Код группы, № специальности, № курса);

R_3 = *Неделя* (№ дня недели, День недели);

R_4 = *Предметы* (№ предмета, Предмет);

R_5 = *Преподаватели* (№ преподавателя, ФИО преподавателя);

R_6 = *Начало занятий* (№ занятия, Время начала занятия);

R_7 = *Оценки* (№ студента, № группы, № предмета, Оценка);

R_5 = *Расписание* (№ группы, № дня недели, № занятия, № предмета, № преподавателя, № аудитории).

Рассматриваемую схему БД учебного заведения можно изобразить в виде Рис. 3., которая является фрагментом БД, представленной на Рис. 1. На

схеме атрибуты первичного ключа отношения помечены символами PK, связи на схеме соответствуют ссылочным ограничениям целостности.

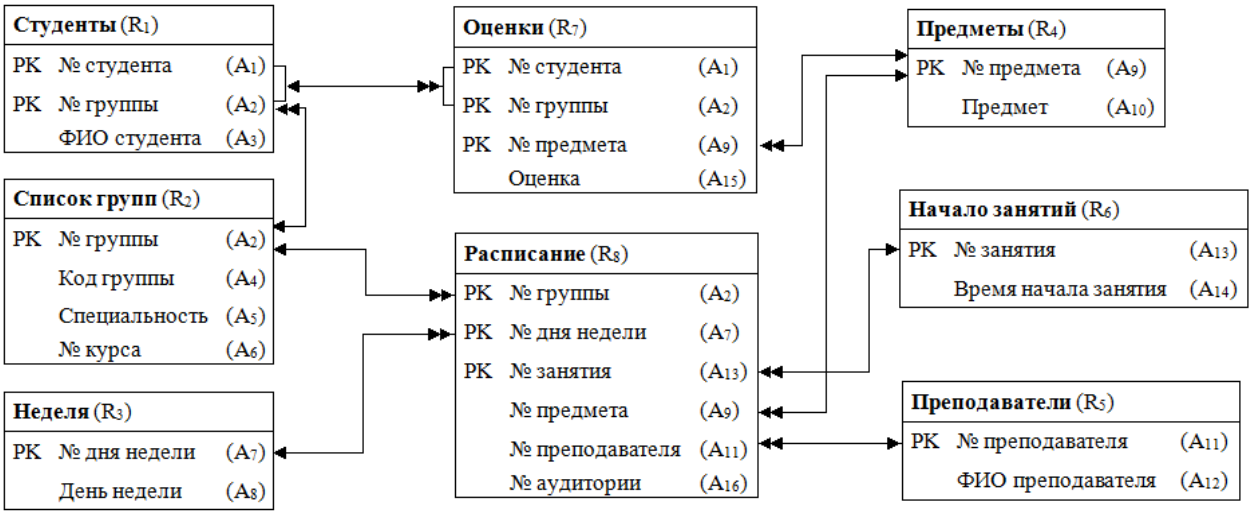

Рис. 3. Фрагмент схемы БД «Вуз».

Именам отношений и атрибутов на схеме БД «Вуз» в скобках поставлены в соответствие символьные обозначения.

3.1 Коммутативные преобразования данных

Для обоснования корректности преобразований традиционно используется [111] свойство коммутативности. В данном разделе приводится свойство коммутативных преобразований, аналогичное работе [111], однако, описание модели данных в данной диссертационной работы было адаптировано для работы с многотабличными представлениями.

Построение межмодельных отображений начинается с выбора, либо с формирования некоторой универсальной модели, которая охватывает возможности по описанию и представлению наиболее широкий класс моделей. В данной работе предполагается, что универсальная модель известна и полностью определена (реляционная). Далее формируется либо выбирается целевая (специализированная) модель данных, с которой и предполагается дальнейшая работа. Чаще всего целевая модель определяется

посредством пользовательского описания данных. Конечной целью построения отображения является разработка программного обеспечения, поддерживающего интерфейс между исполняющей средой (программным обеспечением) для универсальной модели и исполняющей средой для целевой модели, в том числе, для модели пользовательского представления. Существенной деталью построения является то, что пользователь должен иметь возможность корректно выполнять весь спектр операций над данными, не выходя за рамки своего приложения.

Модель информационной системы (*модель данных*) запишем в виде алгебраической системы:

$$\Omega = \langle M, D, O, P \rangle,$$

где M – логическая схема данных; D – совокупность допустимых состояний БД; O – набор операций для модели Ω; P – совокупность предикатов, ограничивающих допустимые состояния D; M и D в совокупности являются носителем системы.

Модель Ω будем считать *исходной*, а $\Omega' = \langle M', D', O', P' \rangle$ – *целевой* (пользовательской). Следовательно, необходимо построение преобразования:

$$\Omega \Rightarrow \Omega'.$$

В [111] рассмотрен метод построения отображения моделей данных, основанный на свойстве коммутативности. В работе [79] получено обобщение метода на случай отсутствия биективности состояний исходной и целевой моделей данных. Такая ситуация является актуальной, когда пользовательское приложение взаимодействует не со всей БД, а только с ее частью. Спецификой данной работы является полностью обновляемое представление данных пользователя при активизации приложения: выполнение многотабличного запроса. Далее рассмотрим

последовательность построения преобразования исходной модели Ω в целевую Ω', при которой гарантируется целостность БД:

1) формируется переход из модели Ω в модель Ω';
2) определяется набор состояний исходной и целевой моделей;
3) формируются ограничения целостности, накладываемые на обе модели;
4) формируется алгоритм преобразования БД, соответствующий пользовательским обновлениям в представлении.

Переход между исходной моделью Ω в целевую Ω' осуществляется при помощи многотабличного запроса, который в свою очередь задается в терминах стандартного языка запросов.

В настоящее время таковым является язык SQL. Запрос будет записан компактнее, если воспользоваться реляционной алгеброй [115, 118], которая является теоретической основой языка SQL. Переход из одного состояния БД в другое осуществляется за счет выполнения операции обновления: вставка, удаление и обновление кортежа в каком-либо отношении.

Определение 3.1. Совокупность всех значений данных в БД, неизменных в течение некоторого промежутка времени, будем называть *элементарным состоянием*.

При работе с моделью Ω' пользователь выполняет команду f'_p, которая переводит модель Ω' из состояния $d'_i \in D'$ в состояние $d'_j \in D'$. В программном обеспечении должны быть реализованы алгоритмы, выполняющие соответствующие преобразования модели Ω из состояния $d_i \in D$ в состояние $d_j \in D$.

Определение 3.2. Преобразование $\Omega \Rightarrow \Omega'$ будем считать *коммутативным*, если выполняются условия: $f'_p\big(Q(d_i)\big) = Q\big(Alg_p(d_i)\big)$,

где Q – многотабличный запрос; Alg_p – алгоритм преобразования исходной

БД, сопоставленный команде обновления f'_p.

Другими словами, корректность выполняемых преобразований можно продемонстрировать на коммутативной диаграмме:

$$\begin{array}{ccc} d'_i & \xrightarrow{f'_p} & d'_j \\ Q \uparrow & & \uparrow Q \\ d_i & \xrightarrow{Alg_p} & d_j \end{array}$$

Представленная диаграмма говорит о том, что из начального состояния d_i в конечное состояние d'_j можно перейти двумя различными способами, при выполнении пользовательских обновлений f'_p над представлением Q, либо при выполнении запроса Q к результату преобразований БД Alg_p. В определении 3.2 предполагается, что при выполнении преобразований состояние БД не изменяется другими приложениями.

Для исходной модели данных Ω может быть задана совокупность ограничений на допустимые состояния. Для реляционной модели данных это, прежде всего, первичные ключи, внешние ключи и ограничения домена для атрибутов. Первичные ключи позволяют реализовать в БД функциональные зависимости [25]. Кроме того, функциональные зависимости могут быть реализованы с помощью уникального индекса, и в крайнем случае, за счет триггера. Теоретической основой для внешних ключей являются зависимости включения [38]. Эти зависимости реализуются за счет связей на схеме БД. Таким образом, ограничения целостности в совокупности определяют допустимые состояния данных и переходы между ними для исходной модели Ω. Поскольку запрос Q определяет соответствующие состояния моделей Ω и Ω', то соответствующие

ограничения на допустимые состояния и переходы модели Ω' также определяются этим запросом. Образы ограничений целостности должны быть достаточными для выполнения исходных ограничений модели Ω: для целевой модели Ω' могут быть заданы более «жесткие» ограничения, если реализация образов ограничений, являющихся необходимыми и достаточными, для модели Ω' является затруднительной либо невозможной. В крайнем случае, ограничением целостности может стать запрет на обновление определенной группы данных.

Выполнение операции преобразования f_p' для модели Ω' должно сопровождаться преобразованием состояния модели Ω. Для этого в системе должен быть реализован алгоритм Alg_p, соответствующий операции f_p' и удовлетворяющий условию коммутативности. Таким образом, для каждой операции f_p' должен быть реализован свой алгоритм преобразования БД: Alg_p. Из этого следует, что набор операций f_p' должен быть строго регламентирован.

При разработке алгоритмов Alg_p основное внимание должно быть уделено корректности преобразований. Однако, выполнение алгоритмов Alg_p предполагается при работе пользователя с приложением (on line), следовательно, должно быть обеспечено приемлемое время их выполнения.

3.2 Теоремы о коммутативности преобразований

В данном разделе приводятся формулы в терминах реляционной алгебры, осуществляющие операции обновления записей в многотабличных представлениях. Формулируются и доказываются теоремы о корректности преобразований. Доказывается теорема о последовательности двух коммутативных преобразований.

Теорема 3.1 Последовательность двух коммутативных преобразований является коммутативным преобразованием.

Доказательство. Пусть пользователь выполняет последовательно операции f_1' и f_2' над представлением Q, тем самым переводя его из состояния d_i' в d_j', затем из состояния d_j' в d_k'. Покажем, что соответствующие операции Alg_1 и Alg_2 над БД будут корректными.

Рассмотрим вторую операцию преобразования

$$d_k' = f_2'\left(Q(d_j)\right) = Q\left(Alg_2(d_j)\right). \quad (3.1)$$

Данная запись говорит о том, что переход из состояния d_j в состояние d_k' возможен двумя различными способами.

Далее рассмотрим формулы для первой операции обновления $d_j' = Q(d_j) = f_1'(Q(d_i))$. Преобразования БД для первой операции обновления: $d_j = Alg_1(d_i)$. Подставляя последние формулы в (3.1) получим следующее уравнение:

$$f_2\left(f_1(Q(d_i))\right) = Q\left(Alg_2(Alg_1(d_i))\right).$$

Слева от знака равенства присутствует последовательность двух пользовательских обновлений f_1 и f_2 над представлением Q. Справа – последовательность двух алгоритмов преобразования БД Alg_1 и Alg_2. В соответствии с определением 3.2 следует, что последовательность двух преобразований БД и последовательность из двух пользовательских обновлений являются коммутативными. *Теорема доказана.*

В данном разделе приводятся аналитические формулы в терминах реляционной алгебры, которые осуществляют преобразования БД в соответствии с обновлением представлений, сделанных пользователем. Формулируются и доказываются теоремы о корректности выполняемых преобразований.

Рассмотрим формальную постановку задачи. Исходная реляционная БД (РБД) представлена в виде множества отношений: $R_1, R_2, \dots, R_k$ Целевая модель – представление, являющаяся результатом выполнения запроса к РБД:

$$Q = \pi_{X_0}\left(\sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])\right),$$

где π_{X_0} – операция проекции по множеству атрибутов X_0, σ_F – операция селекции, F – логическое выражение на атрибутах отношений $R'_1, R'_2, \dots, R'_m$, $\bowtie$ – операция естественного соединения отношений РБД, $R'_i[X_i]$ – краткая запись операции проекции отношения R'_i по атрибутам X_i. Атрибут A_j отношения R'_i принадлежит $X_i, i = 1,2 \dots m$: если выполняется хотя бы одно из следующих условий:

1) $A_j \in X_0$;
2) существует $R'_l, l \neq i$: $A_j \in \langle R'_l \rangle$;
3) $A_j \in \langle F \rangle$,

других атрибутов в X_i нет.

Рассмотрим соответствие обозначений. Схема данных исходной модели данных есть множество схем отношений БД: $M = \{\langle R_1 \rangle, \langle R_2 \rangle, \dots, \langle R_k \rangle\}$, где $\langle R_i \rangle$ – все атрибуты отношения R_i; $M' = \cup_{i=1}^{m} \langle R'_i \rangle$ – заголовок отношения, соответствующий результату выполнения запроса Q; P – первичные и внешние ключи БД; P' – реализованные зависимости БД (образы первичных и внешних ключей); O и O' – базисные для БД операции вставки, удаления и обновления кортежей в отношениях. Допустимые состояния D и D' определяются предикатами P и P' соответственно.

Условие коммутативности для запроса Q интерпретируется следующим образом: если пользователь удалил, добавил или обновил запись,

то после повторной загрузки данных именно эта запись должна быть удалена, добавлена или обновлена. Остальные записи в представлении должны остаться без изменений с точностью до порядка их следования.

Рассмотрим требования к составу и структуре запроса Q.

1) Отношения $R'_1, R'_2, \ldots, R'_m$ должны иметь упорядочение по внешним ключам: главные отношения в последовательности стоят раньше, подчиненные – позже. Таким образом, должен существовать частичный порядок, в котором есть только одно отношение R'_m, не имеющее подчиненных отношений. Это отношение будет соответствовать семантике приложения: выполненные операции в приложении будут реализовываться только соответствующими операциями в R'_m. Далее это отношение будем называть *целевым*. Фактическое установление частичного порядка над отношениями не является принципиальным, однако, алгоритмическая реализация преобразования БД в виде схемы итерирования отношений [110] в большинстве случаев позволяет существенно сократить промежуточные результаты за счет внешних ключей и, как следствие, добиться сокращения времени работы алгоритма.

Сформулируем условия (ограничения) на атрибуты запроса Q. Для корректного выполнения операций должны быть выполнены следующие условия:

2) $\langle R'_m \rangle \to X_0$: атрибуты отношения R'_m функционально определяют атрибуты приложения X_0,

3) $\langle R'_m \rangle \subseteq X_0 \cup X_1 \cup \ldots \cup X_{m-1}$ – отсутствие свободных атрибутов в целевом отношении.

Представленные условия далее должны быть реализованы в инструментальной исполняющей среде при генерации пользовательских многотабличных приложений. С использованием БД Рис. 1 могут быть

сгенерированы представления для различных приложений: «Ведомость на экзамен», «Сводная ведомость по предмету», «Сводная ведомость для группы», «Расписание занятий для группы», «Расписание занятий на день недели» и т.п. Каждая запись представления для перечисленных приложений имеет ровно один кортеж прообраз в целевом отношении, что упрощает процесс обновления. Рассмотрим более сложный пример приложения «Расписание занятий по специальности», в котором одной записи представления соответствует несколько кортежей целевого отношения.

Пример 2. С использованием обозначений [118] запрос на расписание занятий по специальности может иметь следующий вид:

$$Q_1 = \pi_{X_0}\left(\sigma_{A_{10}=\text{Предмет}}(R_2[A_2A_5A_6] \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6 \bowtie R_8)\right),$$

где $X_0 = A_5A_6A_8A_{14}A_{12}A_{16}$. При старте приложения значение параметра 'Предмет' выбирается из списка предметов (отношение R_4). С помощью приложения, в основе которого используется представление Q_1, должна быть возможность дополнения и удаления занятия. Кроме того, должно быть доступно: перенос занятия на другое время и/или день недели, замена преподавателя, замена аудитории.

Операция удаления записи

Пусть в приложении запись u удалена из представления, соответствующего запросу Q. Далее запрос и результат его выполнения будем обозначать одним и тем же символом Q.

В соединении отношений

$$R_1'[X_1] \bowtie R_2'[X_2] \bowtie \cdots \bowtie R_m'[X_m]$$

множество кортежей T, удовлетворяющих логическому выражению F и принимающих значения записи u на атрибутах X_0, можно выразить формулой:

$$T = \sigma_{F\wedge(X_0=u)}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m]),$$

где выражение $(X_0 = u)$ означает равенство значений одноименных атрибутов. Тогда операция удаления записи u из представления Q сводиться к удалению кортежей

$$R''_m = R'_m \backslash \pi_{\langle R'_m\rangle}(T) \tag{3.2}$$

$\pi_{\langle R'_m\rangle}\left(\sigma_{F\wedge(X_0=u)}\left(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m]\right)\right)$ в отношении R'_m, где символом \ помечена реляционная операция вычитания [118]. Покажем, что данное преобразование будет корректным (коммутативным). Это означает, что представление

$$Q' = \pi_{X_0}\left(\sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R''_m[X_m])\right)$$

будет отличаться от представления Q только отсутствием одной записи u. Заметим, что при выполнении операции проекции дублированные кортежи удаляются.

Теорема 3.2. Операция удаления записи $u \in Q$ коммутативна.

Доказательство. 1) Покажем, что запись u будет отсутствовать в Q'. Допустим обратное: $u \in Q'$. Следовательно, существует кортеж t:

$$t \in \sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R''_m[X_m])$$

и $t[X] = u$. Поскольку $R''_m[X_m] \subseteq R'_m[X_m]$, то

$$t \in \sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])$$

и, следовательно, запись должна быть удалена, что противоречит предположению.

2) Покажем, что не произошло удаление лишних записей: если $u' \in Q$ и $u' \neq u$, тогда $u' \in Q'$. Предположим обратное: запись u' отсутствует в Q'. Следовательно, любой кортеж $t' \in \sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])$, для которого $t'[X_0] = u'$, должен удовлетворять условию для удаления

$t'[\langle R'_m \rangle] \in T$. Однако, в T содержатся только те кортежи, для которых существует $t \in \sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])$ и $t[X_0] = u$. Следовательно, для любого кортежа t' найдется кортеж t, для которых $t'[\langle R'_m \rangle] = t[\langle R'_m \rangle]$ и $t'[X_0] \neq t[X_0]$, что противоречит ограничению $\langle R'_m \rangle \rightarrow X_0$. Полученные противоречия доказывают коммутативность операции удаления кортежа. *Теорема доказана*.

Аналитическое выражение (3.2) является замещением алгоритма Alg_1 для удаления кортежа в многотабличном запросе. В исполняющей среде это выражение достаточно для формирования команды SQL с последующим ее выполнением посредством какой-либо СУБД. Однако, не гарантируется эффективное выполнение этой команды, в том числе, по причине значительного объема промежуточных данных. В исполняющей среде предлагается реализовать алгоритм итерирования отношений с динамическим выбором последовательности просматриваемых отношений [110].

Для иллюстрации примера удаления записи из многотабличного представления воспользуемся примером 1. Пусть множество X_0 будет состоять из атрибутов:

1) <Специальность(A_5)>;
2) <№ курса(A_6)>;
3) <День недели(A_8)>;
4) <Время начала занятия(A_{14})>;
5) <ФИО преподавателя(A_{12})>;
6) <№ аудитории(A_{16})>.

В качестве дополнительного ограничения на значения атрибутов выбирается Предмет = ‘Введение в специальность’. Пусть пользователь попытается удалить запись

$u =$ (Физика, 1, Вторник, 11-40, Иванов И.И., Б-314).

После чего аппарат коммутативных преобразований должен сформировать множество $T -$ множество удаляемых записей из целевой таблицы Расписание. На данное множество атрибутов накладывается ограничение

$$F \wedge (X_0 = u),$$

которое рассматривается как конъюнкция условий отбора F:

$$F = \{\text{Предмет } = \text{‘Введение в специальность’}\}$$

и условий $X_0 = u$, которое обозначает равенство значений удаляемой строки u и соответствующих атрибутов в БД. Данное условие формируется как конъюнкция условий:

$$< \text{Специальность}(A_5) > = \text{'Физика'}$$

$$< \text{№ курса}(A_6) > = 1$$

$$< \text{День недели}(A_8) > = \text{'Вторник'}$$

$$< \text{Время начала занятия}(A_{14}) > = \text{'}11 - 40\text{'}$$

$$< \text{ФИО преподавателя}(A_{12}) > = \text{' Иванов И. И.'}$$

$$< \text{№ аудитории}(A_{16}) > = \text{' Б} - 314\text{'}$$

Множество кортежей, подлежащих удалению из целевого отношения представлено в виде:

$$(A_2 = 15, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = \text{'Б} - 314\text{'}),$$

$$(A_2 = 16, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = \text{'Б} - 314\text{'}),$$

$$(A_2 = 17, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = \text{'Б} - 314\text{'}).$$

После чего необходимо отправить на сервер текст транзакции, которая произведет операции удаления необходимого множества записей из целевого отношения.

Операция вставки записи

По аналогии с предыдущей операцией рассмотрим аналитическое выражение операции вставки записи в терминах реляционной алгебры. Пусть запись u вставляется в представление Q, тогда в исходной БД должны быть выполнены следующие преобразования:

$$R''_m = R'_m \cup T \tag{3.3}$$

где:

$$T = \pi_{X_m}\big(\sigma_F(T' \bowtie u)\big),$$

$$T' = \pi_Y\Big(\sigma_{(Z=u[Z])\&F'}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_{m-1}[X_{m-1}])\Big),$$

$$Z = X_0 \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}),$$

$$Y = (\langle R'_m\rangle \cup \langle F\rangle \cup X_0) \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}).$$

если $Z = \emptyset$, то будем считать, что формула $Z = u[Z]$ принимает значение «Истина», F' – проекция формулы F на подпространство атрибутов $\langle F'\rangle = \langle F\rangle \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1})$ [59]. В случае операции вставки решение может отсутствовать: 1) может оказаться пустым множество T' – отсутствуют кортежи, с которыми может быть выполнено соединение (формула $Z = u[Z]$ принимает значение «Ложь» для всех кортежей в T'); 2) может оказаться пустым множество T – отсутствуют кортежи, на которых F принимает значение «Истина».

Покажем, что данное преобразование будет корректным (коммутативным), если $T' \neq \emptyset$ и $T \neq \emptyset$. Это означает, что представление

$$Q' = \pi_{X_0}\big(\sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_{m-1}[X_{m-1}] \bowtie R''_m[X_m])\big)$$

будет отличаться от представления Q только присутствием одной записи u.

Теорема 3.3. Если $T' \neq \emptyset$ и $T \neq \emptyset$, то операция вставки записи $u \in Q$ коммутативна.

Доказательство. Все записи Q будут принадлежать Q', поскольку в отношениях $R_i', i = 1 \dots m$, кортежи не были удалены. Покажем, что запись u, и только она, будет добавлена в Q'. Предварительно заметим, что $\langle R_m' \rangle = X_m$, то есть все атрибуты целевого отношения R_m' участвуют в реализации запросов Q и Q'. Допустим обратное, существует атрибут $A_j \in \langle R_m' \rangle$ и $A_j \notin X_m$. Тогда по правилам формирования X_i должно быть выполнено $A_j \notin X_i, i = 1 \dots m-1$. Из ограничения 3 следует, что $A_j \in X_0$, но такие атрибуты должны быть включены в X_m по построению. Следовательно, $\langle R_m' \rangle \subseteq X_m$. Включение в обратную сторону следует из свойства операции проекции. Таким образом, имеем:

$$X_m \subseteq X_0 \cup X_1 \cup X_2 \cup \dots \cup X_{m-1}$$

и

$$X_0 \subseteq X_1 \cup X_2 \cup X_3 \cup \dots \cup X_m.$$

С учетом полученных неравенств все кортежи T' на атрибутах Z совпадают с кортежем $u[Z]$ по построению. При формировании T кортежи по атрибутам $X_0 \backslash Z$ будут равны значениям соответствующих атрибутов $u[X_0 \backslash Z]$. По свойствам операций реляционной алгебры имеем:

$$Q' = \pi_{X_0}\big(\sigma_F(R_1'[X_1] \bowtie R_2'[X_2] \bowtie \cdots \bowtie R_{m-1}'[X_{m-1}] \bowtie R_m''[X_m])\big) =$$

$$= \pi_{X_0}\Big(\sigma_F\big(R_1'[X_1] \bowtie R_2'[X_2] \bowtie \cdots \bowtie R_{m-1}'[X_{m-1}] \bowtie (R_m'[X_m] \cup T)\big)\Big) =$$

$$= Q \cup \pi_{X_0}\big(\sigma_F(R_1'[X_1] \bowtie R_2'[X_2] \bowtie \cdots \bowtie R_{m-1}'[X_{m-1}] \bowtie T)\big).$$

Множество кортежей $\sigma_F(R_1'[X_1] \bowtie R_2'[X_2] \bowtie \cdots \bowtie R_{m-1}'[X_{m-1}] \bowtie T)$ по предположению не пусто, и все кортежи этого множества по атрибутам

X_0 совпадают с записью u. Следовательно, представление Q' будет отличаться от Q только наличием одной и только одной записи u. *Теорема доказана*.

Замечание. Для выполнения условия коммутативности в формуле (3.3) к R'_m достаточно добавить только один кортеж из множества T, однако, необходимо вставка всех кортежей множества T, что соединит вновь введенную информацию со всей существующей информацией в БД. Это обеспечивается выбором значений свободных атрибутов $X_1 \cup X_2 \cup X_3 \cup ... \cup X_m \backslash Z$ при формировании T'.

Как и в случае с операцией удаления записи, полученное аналитическое выражение (3.3) является замещением алгоритма Alg_2 для вставки записи в многотабличное представление. В исполняющей среде формируется команда SQL, соответствующая формуле (3.3).

Для иллюстрации примера вставки записи в многотабличное представление, воспользуемся примером 1. Пусть множество X_0 будет состоять из атрибутов:

1) <Специальность(A_5)>;
2) <№ курса(A_6)>;
3) <День недели(A_8)>;
4) <Время начала занятия(A_{14})>;
5) <ФИО преподавателя(A_{12})>;
6) <№ аудитории(A_{16})>.

В качестве дополнительного ограничения на значения атрибутов выбирается Предмет = ‘Введение в специальность’. Пусть пользователь попытается вставить запись

$$u = (\text{Химия, 1, Среда, 9-45, Петров П.П., А-212}).$$

После чего аппарат коммутативных преобразований должен

сформировать множество $T-$ множество вставляемых записей в целевую таблицу Расписание. На данное множество атрибутов накладывается ограничение. Однако, необходимо отметить, что множество кортежей T отсутствует в целевой таблице, следовательно, необходимо пройтись по всем отношениям, участвующим в создании представления и собрать множество T', которое можно проинтерпретировать как представление, содержащее все записи, которые удовлетворяют условиям вставляемой записи

$$Z = u[Z],$$

Данное условие формируется как конъюнкция условий:

$$< \text{Специальность}(A_5) > = \text{'Химия'}$$

$$< \text{№ курса}(A_6) > = 2$$

$$< \text{День недели}(A_8) > = \text{'Среда'}$$

$$< \text{Время начала занятия}(A_{14}) > = \text{'9} - 45'$$

$$< \text{ФИО преподавателя}(A_{12}) > = \text{' Петров П. П.'}$$

$$< \text{№ аудитории}(A_{16}) > = \text{' A} - 212'$$

Множество кортежей, подлежащих вставке в целевое отношение, представлено в виде:

$$(A_2 = 22, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 23, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 24, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 25, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212').$$

После чего необходимо отправить на сервер текст транзакции, которая произведет операции вставки необходимого множества записей в целевое отношение.

Операция обновления записи

В общем случае операция обновления записи сводится к операциям удаления старой записи u и вставки новой u'. Алгоритму Alg_3 (операция обновления) соответствует следующее выражение реляционной алгебры:

$$R''_m = R'_m \backslash \pi_{\langle R'_m \rangle}\left(\sigma_{F\wedge(X_0=u)}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])\right) \cup$$

$$\cup\, \pi_{X_m}\left(\sigma_F\left(\pi_Y\left(\sigma_{Z=u'[Z]\&F'}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_{m-1}[X_{m-1}])\right) \bowtie u'\right)\right),$$

где:

$$Z = X_0 \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}),$$

$$Y = (\langle R'_m \rangle \cup \langle F \rangle \cup X_0) \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}).$$

Применив эквивалентные преобразования, получим:

$$R''_m = R'_m \backslash \pi_{X_m}\big(\sigma_F(T \bowtie u)\big) \cup \pi_{X_m}\big(\sigma_F(T \bowtie u')\big), \qquad (3.4)$$

где:

$$T = \pi_Y\left(\sigma_{(Z=u[Z])\vee(Z=u'[Z])\ \&F'}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_{m-1}[X_{m-1}])\right),$$

$$Z = X_0 \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}),$$

$$Y = (\langle R'_m \rangle \cup \langle F \rangle \cup X_0) \cap (X_1 \cup X_2 \cup \ldots \cup X_{m-1}).$$

В формуле (3.4) использованы обозначения формулы (3.3) и следующие преобразования. Выражение

$$\sigma_{F\wedge(X_0=u)}(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie R'_m[X_m])$$

было заменено на эквивалентное

$$\sigma_F(R'_1[X_1] \bowtie R'_2[X_2] \bowtie \cdots \bowtie u).$$

В T объединены списки удаляемых и вставляемых кортежей. Если какие-либо кортежи в T принадлежат обоим спискам, то при формировании $T \bowtie u$ и $T \bowtie u'$ каждый кортеж участвует только в своей операции и лишние

кортежи в T игнорируются. Заметим, если $u' = \emptyset$, то формула (3.4) эквивалентна операции удаления записи u, если $u = \emptyset$, то (3.4) – вставка записи u'.

Теорема 3.4. Операция обновления записи u в представлении коммутативна.

Доказательство. При применении теорем 3.1, 3.2 и 3.3 получаем, что операция обновления представления, которая сводится к удалению старого кортежа u и вставке нового кортежа u' является корректной. *Теорема доказана*.

В большинстве приложений каждая запись представления имеет один кортеж прообраз в целевом отношении, также достаточно выполнить одну команду UPDATE.

Пример 3. Для демонстрации рассмотрим операцию обновления записи в примере 1. Пусть при старте приложения выбрано ограничение на название предмета 'Введение в специальность'. Схема представления (заголовок) следующая:

<Специальность(A_5)>, <№ курса(A_6)>, <День недели(A_8)>, <Время начала занятия(A_{14})>, <ФИО преподавателя(A_{12})>, <№ аудитории(A_{16})>.

Пусть в представлении запись

(Физика, 1, Вторник, 11-40, Иванов И.И., Б-314)

заменили записью

(Химия, 1, Среда, 9-45, Петров П.П., А-212).

Исходной записи в представлении будет соответствовать три кортежа (три группы на специальности 'Физика'):

$$(A_2 = 15, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = 'Б - 314'),$$

$$(A_2 = 16, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = 'Б - 314'),$$

$$(A_2 = 17, A_7 = 2, A_{13} = 3, A_9 = 6, A_{11} = 362, A_{16} = 'Б - 314').$$

В соответствии с предложенным подходом в целевом отношении R_8 перечисленные кортежи должны быть заменены четырьмя кортежами (четыре группы на специальности 'Химия'):

$$(A_2 = 22, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 23, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 24, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212'),$$

$$(A_2 = 25, A_7 = 3, A_{13} = 2, A_9 = 6, A_{11} = 412, A_{16} = 'A - 212').$$

Если бы значения атрибута A_2 совпадали для всех новых кортежей, было бы достаточно одной команды UPDATE для выполнения замены. Однако, для данного примера самым эффективным способом является выполнение двух команд SQL: удаление (DELETE) старых кортежей и вставка (INSERT) новых.

3.3 Сопроцессор коммутативных преобразований базы данных

Сопроцессор коммутативных преобразований (СКоП) – информационная система, предназначенная для обновления многотабличных представлений данных. Основными операциями, осуществляемыми СКоП, являются операции удаления, добавления и обновления записи в представлении.

Для того, чтобы получить возможность обновления представлений, СУБД должна корректно выполнять операции, выраженные аналитическими формулами 3.2, 3.3 и 3.4. Для этого совместно с СУБД на рабочей станции пользователя должна работать информационная система, которая имеет архитектуру, представленную на Рис. 4.

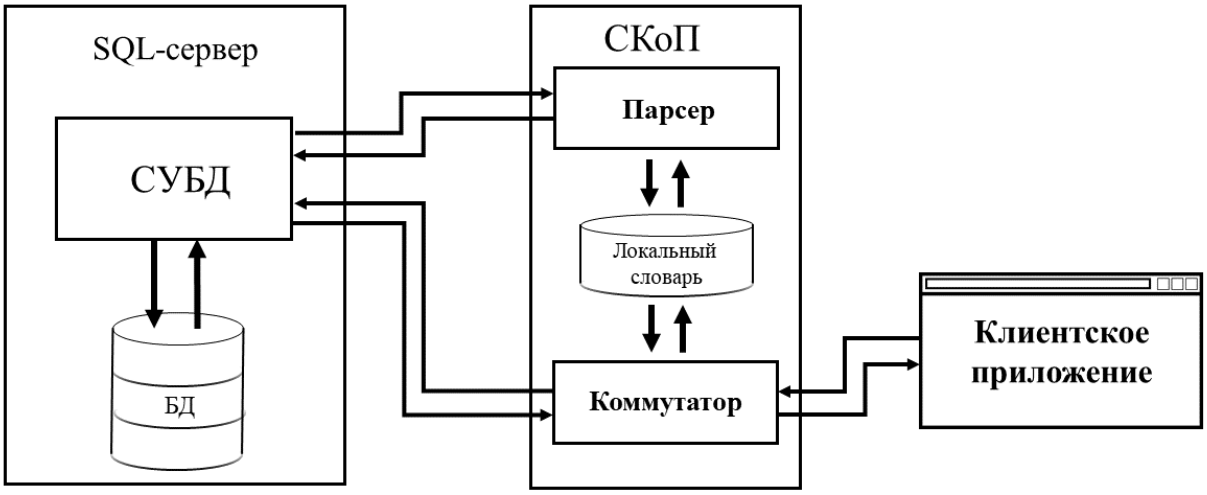

Рис. 4. Архитектура системы БД с Сопроцессором коммутативных преобразований

Архитектура СКоП предполагает наличие трех основных подсистем: Коммутатора, Парсера и Локального каталога. Сопроцессор связывается с сервером через Парсер и Коммутатор. Пользователь посредством Клиентского приложения взаимодействует с Коммутатором.

Парсер – это подсистема, предназначенная для получения метаданных из системных отношений БД, их приведения к специальному виду и сохранения в Локальном каталоге. Основные функции Парсера представлены в Лист. 3 Функция GetListOfTables выгружает из системных отношений БД наименования отношений и записывает их в Локальный каталог. Функция GetListOfAttrs предоставляет набор атрибутов из указанного в качестве аргумента отношения. Функция GetSourceTable среди всего набора отношений по внешним ключам определяет целевое отношение.

Локальный каталог в рассматриваемом Сопроцессоре обеспечивает хранение метаданных. Здесь в специальном виде содержится информация о схеме данных: наименования отношений, неизбыточные множества атрибутов для каждого отношения, а также логические выражения, накладываемые на значения атрибутов. Локальный каталог представляет собой набор csv файлов, которые при инициализации клиентского приложения загружаются

в оперативную память. Запись в локальном каталоге обновляется при обновлении многотабличного представления в БД.

Лист. 3: Интерфейс Парсера

```
//Получение набора доступных пользователю отношений из БД.
//Tables - список отношений.
//Возвращает количество отношений, записанных в поле Tables или отрицательный код ошибки.
int GetListOfTables ( string[] Tables )

//Получение набора атрибутов из указанного отношения.
//TableName - наименование рассматриваемого отношения,
//Attrs - набор атрибутов.
//Возвращает количество записанных атрибутов, или отрицательный код ошибки.
int GetListOfAttrs ( string TableName, string[] Attrs )

//Получение строки оператора WHERE, соответствующего операции естественного соединения.
//WHEREstr - строка SQL оператора WHERE,
//Tables - список отношений, которые будут использоваться для создания представления.
//Возвращает 1 в случае успеха, или отрицательный код ошибки.
int GetWHEREstring ( string WHEREstr, string[] Tables )

//Поиск целевого отношения.
// SourceTable - наименование целевого отношения.
//Tables - список отношений, которые будут использоваться для создания представления.
//Возвращает 1 в случае успеха, или отрицательный код ошибки.
int GetSourceTable ( string SourceTable, string[] Tables )
```

Коммутатор – подсистема, которая формирует транзакцию на обновление набора кортежей в целевом отношении БД. В этой подсистеме используются данные из Локального каталога, реализуются операции реляционной алгебры, формирующие множества кортежей, подлежащих удалению и вставке в целевое отношение. Далее реализуются операции 3.2, 3.3 и 3.4 в

Лист. 4: Интерфейс Коммутатора

```
//Формирование множества кортежей из целевого отношения, подлежащих удалению.
//Tuples - список кортежей.
//Возвращает количество удаляемых кортежей, или отрицательный код ошибки.
int GetTuplesToDELETE ( string[] Tuples )

//Формирование множества кортежей из целевого отношения, подлежащих вставке.
//Tuples - список кортежей.
//Возвращает количество кортежей, подлежащих вставке, или отрицательный код ошибки.
int GetTuplesToINSERT ( string[] Tuples, string query )

//Формирование строки SQL транзакции на удаление кортежей из целевого отношения.
//Tuples - набор кортежей, подлежащих удалению.
//Возвращает 1 в случае успеха, или отрицательный код ошибки.
int FormDELETEquery ( string[] Tuples )

//Формирование строки SQL транзакции на вставку кортежей в целевое отношение.
//Tuples - набор кортежей, подлежащих вставке,
//query - SQL транзакция для сервера БД.
//Возвращает 1 в случае успеха, или отрицательный код ошибки.
int FormINSERTquery ( string[] Tuples, string query )

//Функция отправки SQL транзакции на сервер БД.
//query - SQL транзакция для сервера БД.
//Возвращает количество обработанных кортежей в отношении, или отрицательный код ошибки.
int ExecuteNonQuery( string query )
```

виде транзакций на обновление отношений и отправляются на сервер. Клиентское приложение получает обратную связь от сервера через Коммутатор, в которой содержится информация о результате выполнения операции обновления, либо сведения об ошибке.

Интерфейс Коммутатора представляет функции для выполнения операций обновления записей в представлении, представленные в Лист. 4. Функция GetTuplesToDELETE выбирает из целевого отношения набор записей, которые соответствуют удаляемой записи в многотабличном представлении. Функция GetTuplesToINSERT формирует в виде представления набор записей, необходимых для вставки в целевое отношение. Функции FormDELETEquery и FormINSERTquery формируют SQL запросы на удаление и вставку записей соответственно. Функция ExecuteNonQuery отправляет на сервер сформированный ранее SQL запрос.

Для того, чтобы разработанные алгоритмы были использованы с СУБД, она должна удовлетворять стандарту [21] и содержать каталоги. В СКоП системные каталоги нужны для того, чтобы получить информацию о схеме: данные об атрибутах, отношениях, внешних ключах. Далее эти данные записываются в Локальный каталог. Операции по выгрузке данных из системных каталогов осуществляются при помощи Парсера.

Алг. 5 DELETE (IN $u, F, R_1, \dots, R_m, X_0, X_1, \dots, X_m$; OUT R_m)

for i **from** 1 **to** m **do**
  for j **from** 1 **to** m **do**
    if $i \neq j$ **then**
      $T_D \leftarrow T_D \cup \pi_{X_i}(R_i) \bowtie \pi_{X_j}(R_j)$
$T_D \leftarrow \pi_{X_m}\left(\sigma_{F \wedge (X_0 = u)}(T_D)\right)$
$R_m \leftarrow R_m \setminus T_D$
return R_m

Рассмотрим основные функции Коммутатора. Процедура удаления записи из представления описана в Алг. 5. В переменной T_D хранятся записи, полученные при использовании функции GetTuplesToDELETE из Лист. 4. Операция $R_m \leftarrow R_m \setminus T_D$ соответствует транзакции Коммутатора ExecuteNonQuery. Представленный алгоритм соответствует аналитической

формуле 3.2 и осуществляет коммутативные операции удаления записи из представления.

Алг. 6 INSERT (IN $u, F, R_1, \dots, R_m, X_0, X_1, \dots, X_m$; OUT R_m)

$Y' \leftarrow \langle R_m \rangle \cup \langle\ F\ \rangle \cup\ X_0$
for { i **from** 1 **to** $m-1$ }
    $Z \leftarrow\ Z \cup (X_0 \cap X_i)$
    $Y \leftarrow Y \cup (Y' \cap X_i)$
    $\langle F' \rangle \leftarrow \langle F' \rangle \cup (\langle F \rangle \cap X_i)$
for { i **from** 1 **to** $m-1$ }
    for { j **from** 1 **to** $m-1$ }
        if $i \neq j$ then
            $T_I' \leftarrow\ T_I' \cup \pi_{X_i}(R_i) \bowtie \pi_{X_j}(R_j)$
$T_I' \leftarrow \pi_Y\left(\sigma_{F' \wedge (Z = \pi_{Z(u)})}(T')\right)$
$T_I \leftarrow \pi_{X_m}(\sigma_F(T_I' \bowtie u))$
$R_m \leftarrow R_m \cup T_I$
return R_m

Также пользователь имеет возможность ввести новую запись в представление. В этом случае в СКоП должен быть сформирован запрос на получение множества T_I, которое может интерпретироваться как множество записей, подлежащих вставке в целевое отношение и которые связаны с вставляемой записью в представление. Процедура добавления новой записи представлена в Алг. 6. Описанная процедура соответствует формуле реляционной алгебры 3.3.

Следует заметить, что в целевом отношении связные с u кортежи отсутствуют, следовательно, при формировании запроса на получение множества T_I необходимо составить множество кортежей T_I', которое можно рассматривать как представление, основанное на отношениях $R_1, R_2, \dots, R_{m-1}$ без целевого $R_m[X_m]$ и совпадает со значениями вставляемой записи u по соответствующим атрибутам. Заголовком представления T_I' будет множество атрибутов $\bigcup_{i=1}^{m-1} X_i$. После сформированного множества записей, кото-

рые связаны с вставляемой записью в представление, формируется множество T_I. Данное представление хранит записи, подлежащие вставке в целевое отношение. Заголовком в T_I является множество атрибутов целевого отношения. В качестве исходных данных для T_I выступают вставляемая запись и множество T_I' , также накладываются ограничения F из исходного представления Q. В интерфейсе Коммутатора представление T_I формируется при использовании функции GetTuplesToINSERT. Транзакция $R_m \leftarrow R_m \cup T_I$ на изменение значений в отношении БД описывается в Лист. 4 как функция ExecuteNonQuery.

Операция обновления записи представления в общем виде сводится к удалению старой записи и добавлению новой. При выполнении этой операции множество записей T_U, с которыми может быть связана запись u, будет содержать дополнительные условия $\left(Z = u[Z]\right) \vee \left(Z = u'[Z]\right)$ при выполнении операции выбора. Данные условия говорят о том, что в T присутствуют записи, которые связываются и со старой записью, и с новой. Далее осуществляется удаление из целевого отношения записей, соответствующих старой записи представления $R_m \backslash \pi_{X_m}\left(\sigma_F(T_D \bowtie u)\right)$ и добавление новой записи $R_m \cup \pi_{X_m}\left(\sigma_F(T_I \bowtie u')\right)$.

3.4 Выводы

Предложенный в разделе 3.2 подход к обновлению многотабличных представлений является основой для нового подхода. Для этого создается программа-шаблон, в которую в диалоговом режиме вставляются компоненты запроса Q. В ходе диалога проверяются ограничения на запрос с использованием реализованных зависимостей. Операции реляционной алгебры в виде команд SQL вставляются в программу-шаблон. В результате получа-

ется готовая к исполнению программа, реализующая обновления многотабличных представлений. Основные результаты, описанные в данной главе, были опубликованы в [78, 101, 104, 108].

ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

В данной главе рассматривается реализация подхода к обновлению многотабличных представлений для свободной СУБД PostgreSQL. Приводятся вычислительные эксперименты, целью которых является подтверждение эффективности сопроцессора по сравнению с работой триггеров в различных СУБД.

4.1 Реализация сопроцессора коммутативных преобразований

Данный раздел содержит описание сопроцессора, реализующего обновление многотабличных представлений на основе коммутативных преобразований данных. Подход может применяться для любых БД под управлением СУБД PostgreSQL. При обновлении данных удовлетворяется условие коммутативности данных, которое рассмотрено в разделе 3.1.

4.1.1 Реализация Сопроцессора для СУБД PostgreSQL

Разработанное программное обеспечение будет полезно администратору БД на любом предприятии, работающем с информационными системами. Однако, целевая аудитория – опытные пользователи, которым предоставляется возможность создания многотабличных представлений данных (на основе запроса) в режиме конструктора. После чего, в следующем окне становится доступен весь спектр операций обновления представления.

Предложенная архитектура Сопроцессора коммутативных преобразований была реализована для свободной СУБД PostgreSQL. Соответствующий Сопроцессор получил название pgCOPCT (сокращенно от PostgreSQL COProcessor Commutative Transformations). Данное приложение было разработано на программной платформе .NET Framework на языке программиро-

вания C# [100]. Программа является оконным приложением под управлением операционной системой Microsoft Windows. Соединяется с БД Postgres, находящейся на Локальной машине или сервере.

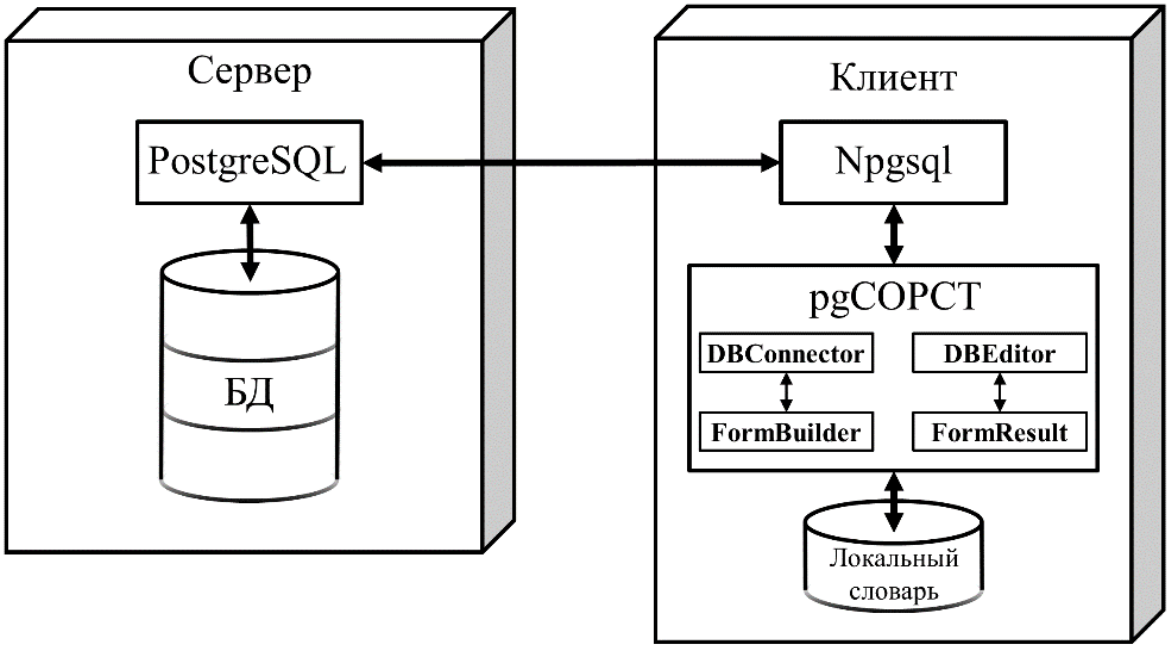

Рис. 5. Структура программы PoVE

Рассмотрим основные компоненты приложения: конструктор запросов (FormBuilder) и редактор представлений (FormResult). Конструктор необходим для формирования многотабличного представления, он обращается к классу DBConnector, который в свою очередь заполняет Локальный каталог, выполняя необходимые запросы к БД. Редактор представлений дает пользователю возможность в сформированном представлении осуществить операции обновления данных. Здесь формируется SQL запрос на обновление данных. Далее редактор представлений посылает сформированную транзакцию на сервер через класс DBEditor. Библиотека Npgsql.dll представляет собой драйвер для подключения СУБД PostgreSQL к среде Visual Studio.

Для того, чтобы реализовать Сопроцессор для другой СУБД, на клиентскую машину необходимо будет загрузить драйвер данных для соответствующей СУБД. Модификации подвергнется только часть Парсера, отвеча-

ющая за выгрузку метаданных из системных отношений СУБД. Объем перерабатываемого исходного кода программы не будет превышать 5%. Из вышесказанного следует, что реализацию СКоП можно считать мобильной и ее можно применить для любой СУБД, удовлетворяющей стандарту [30].

4.1.2 Пользовательский интерфейс приложения pgCOPCT

При запуске программы производится авторизация в системе. Пользователю открывается окно, представленное на Рис. 6. Здесь указываются данные самого пользователя, такие как «Имя пользователя», «Пароль», а также данные для создания соединения: адрес сервера, порт и наименование БД, над которой будут осуществляться операции обновления.

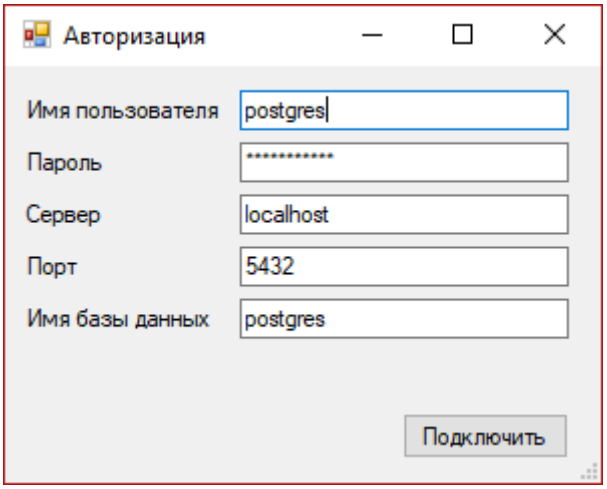

Рис. 6. Окно авторизации в системе

После ввода всех данных для входа в систему администратору БД предоставляется возможность создания многотабличного представления данных в окне конструктора запросов (Рис. 7). При загрузке этой формы из БД выгружается список всех отношений и помещается в левое нижнее окно. В среднее окно «Выбранные таблицы» пользователь переносит из левого окна необходимые отношения для создаваемого запроса путем двойного нажатия на левую кнопку манипулятора мыши. В правом окне находятся все атрибуты из одного текущего отношения. На их основе формируется заголо-

вок представления при помощи двойного щелчка мыши. Выбранные атрибуты помещаются в виде столбцов в верхнее поле. Здесь можно ввести дополнительные ограничения на значения атрибутов и необходимость вывода в заголовок запроса.

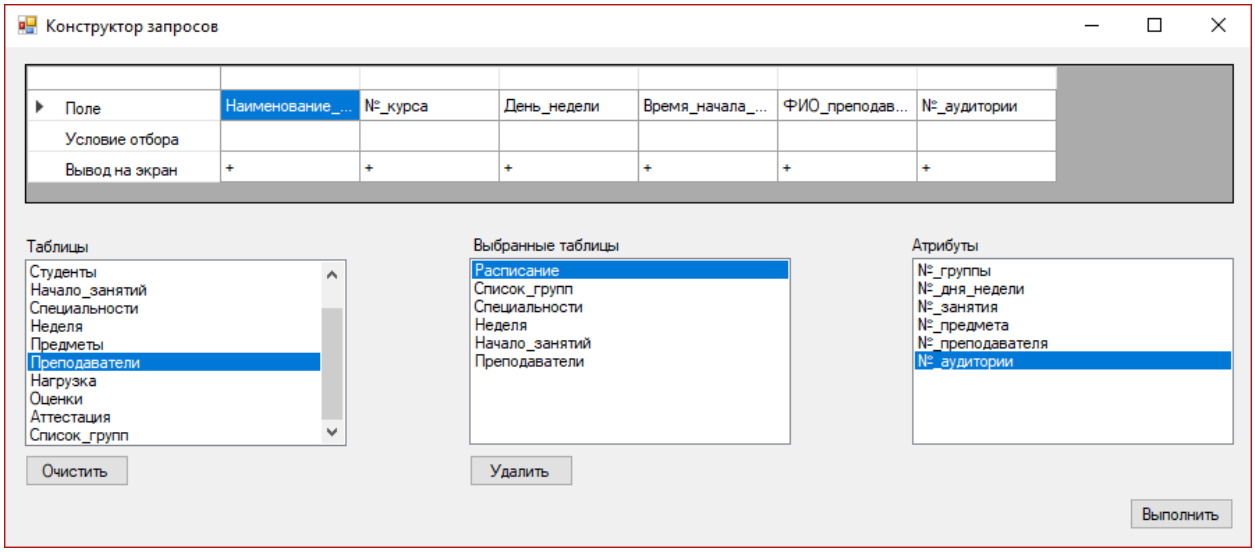

Рис. 7. Конструктор запросов

После формирования всего представления и нажатия на кнопку «Выполнить» приложение открывает следующее окно редактора представлений (Рис. 8).

На Рис. 8 в верхней части формы представлен результат выполненного запроса к БД. В нижнем левом углу указано целевое отношение, в котором будут производиться операции обновления. Необходимо заметить, что целевое отношение определяется автоматически, путем перебора всех внешних ключей. При изменении значения атрибута, который не является компонентом целевого отношения, например, «Наименование_специальности» из отношения «Специальности», в левом нижнем окне высветятся возможные значения для данного атрибута, взятые из отношения-справочника. При двойном нажатии левой кнопки мыши по указанному значению оно помещается в соответствующее поле представления. Если атрибут в представлении является компонентом целевого отношения, то его

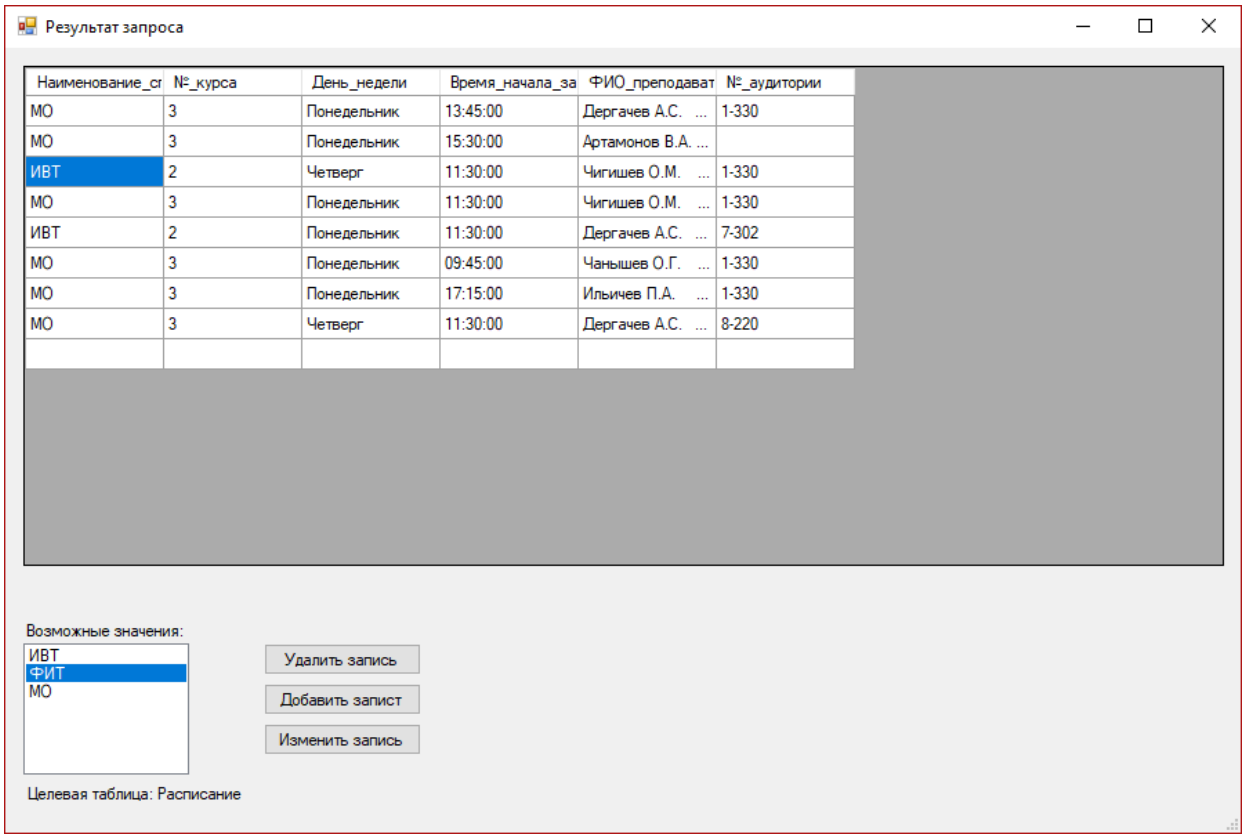

Рис. 8. Многотабличное представление данных

значение вводится вручную с клавиатуры. По завершении обновления одной записи в представлении пользователь нажимает на одну из представленных кнопок, соответствующую необходимой операции. Стоит отметить, что добавление новой записи возможно только в последнюю строчку представления. После выполнения операции обновления представления на экран выводится справка, представленная на Рис. 9, которая содержит информацию о количестве измененных записей в целевом отношении.

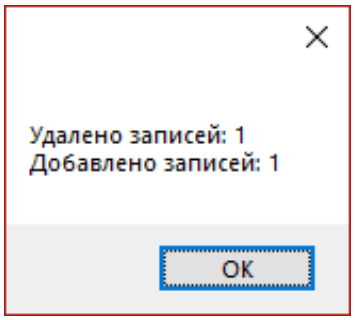

Рис. 9. Справка о выполнении операции обновления

В случае попытки совершить изменения, не удовлетворяющие условиям ссылочной целостности, программа выведет сообщение, полученное от СУБД, где указывается вся информация о возникшем исключении.

4.1.3 Описание событий приложения pgCOPCT

Приложение PoVE разработано в среде программирования MicroSoft Visual Studio при использовании технологии событийного программирования.

Далее рассмотрим составляющие компоненты программы:

1. Оконная форма Authorization – выполняет функцию формирования строки подключения к БД. Вызывается из события загрузки формы Form_Builder
2. Оконная форма FormBuilder – содержит в себе все данные для конструктора запросов (рисунок 4) и обрабатывает все события присущие формированию представления.
    a. Событие FormBuilder_Load – является точкой входа в исполняемую часть программы. Происходит во время загрузки формы. В течении этого события происходит вызов окна авторизации, после чего добавляет в поле «Таблицы» все отношения, доступные из БД;
    b. Событие lbSource_DoubleClick вызывается вследствие двойного клика манипулятором мыши на поле «Таблицы». При этом осуществляется перенос выбранных отношений в поле «Выбранные таблицы», на основе которых строится многотабличное представление данных.
    c. Событие btTableRemove_Click происходит вследствие нажатия кнопки «Удалить» и производит удаление выбранного отношения из поля «Выбранные таблицы».
    d. Событие lbChosen_DoubleClick вызывается после двойного клика на поле «Выбранные таблицы». Данное событие заполняет поле

«Атрибуты» атрибутами из выбранного отношения, на котором было произведено данное событие.

e. Событие lbAttributes_DoubleClick происходит вследствие двойного нажатия на поле «Атрибуты» и осуществляет формирование заголовка будущего представления.
f. Событие btClear_Click вызывается после нажатия на кнопку «Очистить» и производит очистку всех полей формы, заново загружает список отношений, используя сохраненную в оперативной памяти строку подключения к БД.
g. Событие btExeute_Click вызывается вследствие нажатия кнопки «Выполнить» При этом осуществляется сбор всех данных для запроса. Происходит инициализация класса DBConnector и формы редактора представлений FormResult.

3. Оконная форма FormResult (рисунок 5) открывается после нажатия кнопки «Выполнить» в окне конструктора запросов. В ней представлен результат выполнения запроса на выборку из БД. Отдельно передается список отношений и иная информация, сформированная на этапе создания запроса.

a. Событие FormResult_Load вызывается вследствие загрузки окна редактора представлений «Результат запроса». При этом происходит заполнение центрального поля данными, полученными в результате запроса на выборку.
b. Событие dgData_Click происходит вследствие одинарного клика манипулятором мыши на поле данных. Если нажатие было произведено на ячейку атрибута внешнего отношения, то поле «Возможные значения» заполняется из отношения-справочника. Возможность вручную изменить значение в ячейке блокируется. Если же атрибут является неключевым в целевом отношении, то поле «Возможные значения» очищается и предоставляется возможность

вручную с клавиатуры ввести новые значения в ячейку поля данных.

На этом этапе осуществляется также запоминание старой записи для дальнейшего использования. В случае невыполнения нажатия на кнопки обновления и повторного клика на другую строку в представлении в текущую строку возвращается исходное значение.

c. Событие lbValues_DoubleClick возникает вследствие двойного нажатия на поле «Возможные значения» и осуществляет перенос выбранного значения атрибута в текущую ячейку в поле данных.
d. Событие dgData_CellValueChanged происходят в тот момент, когда изменяется значение в ячейке поля данных. При этом происходит проверка введенных значений с дополнительными условиями на значения атрибутов, которые были заданы в конструкторе.
e. События btDelete_Click, tbAddClick и btUpdate_Click возникают при нажатии на кнопки «Удалить запись», «Добавить запись» и «Изменить запись» соответственно. При этом осуществляется необходимая операция обновления представления в соответствии с рассмотренным подходом.
f. Функции removeSpaces(string inputStr) и removeSpaces(string inputStr, bool isForQuery) принимают значения каждой ячейки и проверяют, какой тип данных принимает значение, если тип строковый, то производится удаление пробелов с конца значения. Эти пробелы возникают при избыточной длине строковой переменной в БД. К примеру, если длина поля составляет 256 символов, а введенное значение занимает только 15 символов, то в значении переменной оставшиеся 241 символ будут соответствовать пробелу.

Данная функция имеет одно переопределение с дополнительным аргументом bool isForQuery. Второй вариант функции выделяет каждую строковую переменную апострофами, вследствие чего это значение можно добавлять в условия запросов.

g. Функция Refresh(DBEditor editor) вызывается после операции обновления значений в БД и производит обновление запроса на выборку в поле данных. Функция принимает в качестве параметра экземпляр класса редактора данных для необходимых операций связи с БД.

4. Класс DBConnector осуществляет выгрузку всей необходимой информации из БД для формы конструктора.
   a. Функция List<string> GetListOfTables() возвращает список отношений, присутствующих в БД.
   b. Функция string GetSourceTable(List<string> tables) возвращает одну строку, в которой хранится наименование целевого отношения. Функция принимает в качестве аргументов список отношений, над которыми будут выполняться операции.
   c. Функция string GetWHEREstring(List<string>tables) возвращает строку, где хранится полное выражение SQL запроса WHERE. В качестве входных данных используется набор отношений.
   d. Функция List<string> GetListOfAttributes(string TableName) возвращает список атрибутов из указанного в качестве параметра отношения.
   e. Функция List<string> GetValuesOfAttribue(string AttName, string TableName) возвращает всевозможные значения указанного атрибута из указанного отношения.
   f. Функция List<string> GetForeignAttributes(string attributeName, string TableName) возвращает список атрибутов из внешнего отношения. В качестве аргументов выступают наименование исходного

отношения и наименование атрибута, по которому устанавливается связь. Если указанный атрибут не является частью внешнего ключа, то функция возвращает пустой список.

g. Функция List<string> GetForeignAttributeValues(string attributeName, string TableName) возвращает значения атрибутов из внешнего отношения для указанного атрибута. В качестве аргументов функция принимает наименование целевого отношения и имя атрибута внешнего ключа, на котором произошло событие dgData_Click.
h. Функция bool isAttributeInTable(string attName, string tableName) проверяет наличие указанного атрибута в указанном отношении. Возвращаются значения либо «истина», либо «ложь».
i. Функция List<string> ExecuteQuery(string query) исполняет указанный в качестве аргумента запрос к БД. В качестве выходных данных возвращаются список записей полученного представления.

5. Класс DBEditor осуществляет перенос данных из БД в форму редактора. При этом присутствуют некоторые аналогичные функции, которые описаны в классе DBConnector. Далее приводится список уникальных функций, присущих только рассматриваемому классу.

a. Функция List<string> GetLinkingAttributes(List<string> tables) возвращает список атрибутов, по которым устанавливаются связи. В качестве аргументов принимается набор выбранных отношений. Данная функция используется для формирования множества $X_1 \dots X_{m-1}$.
b. Функция List<string> ReplaceFKwithPK(List<string> attributes, List<string> tables) заменяет наименования атрибутов внешнего ключа в дочернем отношении атрибутами первичного ключа отношения-справочника. Данная операция имеет значение, когда связываются два отношения по разноименным атрибутам.

c. Функция int ExecuteNonQuery(string query) отправляет запрос на изменение данных средствами СУБД. В качестве возвращаемого значения выступает числовая переменная, содержащая информацию о количестве обработанных записей.

Для выполнения операций над БД используются системные отношения из каталога pg_catalog.

- Отношение pg_class хранит в виде данных наименования отношений пользовательской БД и их идентифицирующие номера.
- Отношение pg_constraint содержит информацию о всех ограничениях целостности. К примеру, для каждого внешнего ключа в данном отношении указываются номера отношений, между которыми устанавливаются связи, а также порядковые номера атрибутов внешнего и первичного ключей.
- Отношение pg_attributes хранит информацию о всех атрибутах в БД, их наименования, номера отношений, к которым они относятся и, в том числе, порядковый номер данного атрибута в соответствующем отношении.

4.2 Вычислительные эксперименты

В данном разделе описаны вычислительные эксперименты, демонстрирующие эффективность предложенных подходов. Под эффективностью понимается время выполнения операций обновления многотабличного представления с помощью сопроцессора коммутативных преобразований в сравнении с современными СУБД PostgreSQL, Oracle и MS SQL Server, которые для выполнения аналогичных преобразований используют триггеры.

Применение СКоП исследовалось для двух классов приложений: OLAP и OLTP. Приложения OLAP (Online Analitical Processing, оперативный

анализ данных) связаны с выполнением сложных запросов на выборку данных из нескольких отношений хранилища данных и с использованием агрегационных функций. В связи с этим для приложений OLAP применение СКоП ограничивается случаем обновления хранилища данных, которому соответствует выполнение операций вставки новых записей в многотабличное представление.

Приложения OLTP (Online Transaction Processing, оперативная обработка транзакций) подразумевают обработку коротких транзакций, которые выполняют вставку, удаление и обновление кортежей в БД в реальном времени. В соответствии с этим для приложений OLTP исследовалась эффективность СКоП при выполнении операций вставки, удаления и обновления записей в многотабличном представлении.

Эксперименты проводились с использованием синтетических БД, созданных в соответствии со спецификациями стандартных тестов консорциума TPC (Transaction Processing Council): TPC-H [26] для приложений класса OLAP и TPC-E [48] для приложений класса OLTP. При этом распределение кортежей в БД осуществлялось по правилу Зипфа «80-20» [7]: 80 % кортежей в целевом отношении соответствует 20 % кортежей в отношении-справочнике.

Аппаратная платформа экспериментов резюмирована ниже в таблице

Тип процессора	Intel Core 2 Duo P7450 (2 ядра по 2.13 ГГц)
Оперативная память	4 Гб (DDR3-533)
Дисковая память	256 Гб, твердотельный накопитель OCZ
Операционная система	MS Windows 10 Pro

4.2.1 Эффективность Сопроцессора в приложениях класса OLAP

В стандартном тесте TPC-H [26] СУБД имитирует обработку заказов, используя БД, схема которой представлена на Рис. 10. Базовыми отношениями многотабличного представления выбраны отношения CUSTOMER, ORDERS, LINEITEM и PARTSUPP. В качестве целевого отношения

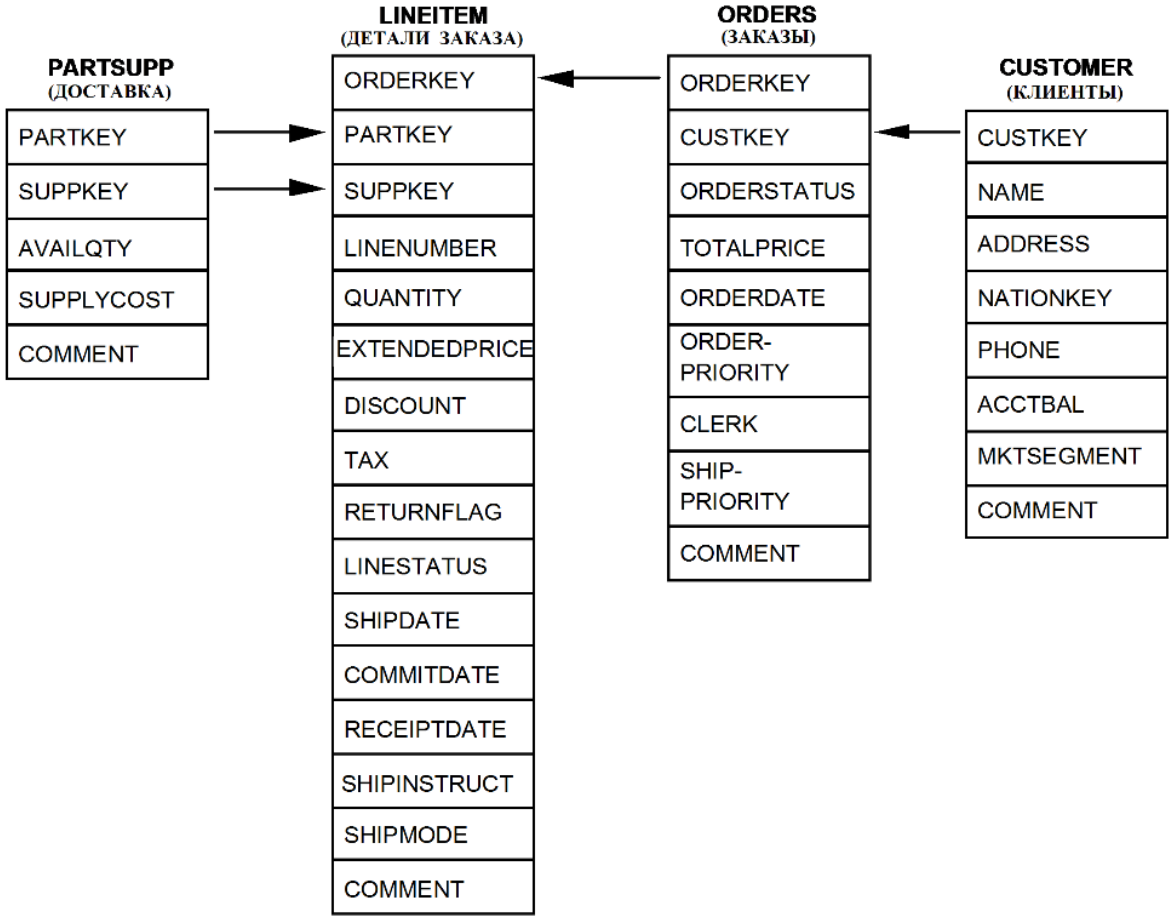

Рис. 10 Схема БД теста TPC-H

выбрано отношение LINEITEM. Для случая хранилищ данных целевое отношение можно интерпретировать как таблицу фактов. Многотабличное представление задействует все атрибуты целевого отношения, не являющиеся атрибутами внешнего ключа, а также атрибуты отношения CUSTOMER.Name, ORDERS.OrderPriority, PARTSUPP.Comment, к которым применено ограничение по атрибуту CUSTOMER.Address (отбор клиентов из специфицированного города). В общем виде сформированное представление можно интерпретировать как поставка для всех клиентов из определенного региона. Многотабличное представление формировалось таким образом, чтобы одна запись в представлении соответствовала N кортежам в целевом отношении. В качестве изменяемого параметра экспериментов используется переменная N.

Для случая приложений OLAP применение СКоП ограничивается сценарием обновления хранилищ данных, в результате которого происходит добавление новых записей в хранилище данных. Следовательно сопроцессор будет выполнять только операции вставки записи в представление, после чего на сервер будет отправляться текст транзакции на добавление новых записей в таблицу фактов.

Прежде чем производить вычислительные эксперименты обновления записи в многотабличном представлении, необходимо произвести обновление Локального каталога. При создании представления, основанного на базовых таблицах Рис. 10, были сделаны запросы к системным таблицам БД, после чего данные были записаны в Локальный каталог. Временные расходы на данную операцию составили 0.045 секунды. В то время как чтение данных из Локального каталога в оперативную память потребовало 0.0029 секунд. Представленные накладные расходы оказывают несущественное влияние на результаты численных экспериментов. На графиках Рис. 11, Рис. 12, Рис. 13 не учтены накладные расходы на обновление Локального каталога и считы-

вание данных из каталога в оперативную память, поскольку они несущественны по сравнению со временем обновления представлений (не более 1%).

Для каждой из трех СУБД, MS SQL Server, ORACLE и PostgreSQL, были созданы триггеры INSTEAD OF, которые ассоциированы с операцией добавления нового кортежа в хранимом представлении VIEW. Для каждого тестового прогона триггер выполнял операцию вставки INSERT INTO, сформированную с помощью Коммутатора в СКоП.

Вследствие того, что сопроцессор был реализован только для СУБД PostgreSQL, для других СУБД необходимо было реализовать только часть Коммутатора, которая отправляет на сервер текст транзакции. Сам текст транзакции был собран сопроцессором для PostgreSQL, после чего перестроен под конкретную СУБД. Таким образом были получены значения для графиков СКоП ORACLE Рис. 12 и СКоП MS SQL Server Рис. 13. Однако необходимо учитывать, что при этом Парсер не выполняет сбор метаданных из системных таблиц конкретной СУБД и на графиках эти данные не учитываются.

Рассмотрим результаты экспериментов для сопроцессора. На Рис. 11 можно видеть, что для случая СУБД PostgreSQL применение сопроцессора уменьшает время выполнения операции обновления многотабличного представления на 35% по сравнению со временем работы триггеров. В СУБД MS SQL Server применение сопроцессора дает преимущество по времени на 17% (см. Рис. 13). Результаты вычислительных экспериментов для СУБД ORACLE представлены на Рис. 12, здесь можно видеть, что Сопроцессор выполняют операции вставки данных в многотабличное представление с помощью СКоП быстрее, чем с помощью триггеров, на 31% быстрее. Из результатов графиков Рис. 11 – Рис. 13 можно видеть, что применение сопроцессора наиболее эффективно при работе с СУБД PostgreSQL, однако механизм триггеров наиболее развит в СУБД MS SQL Server.

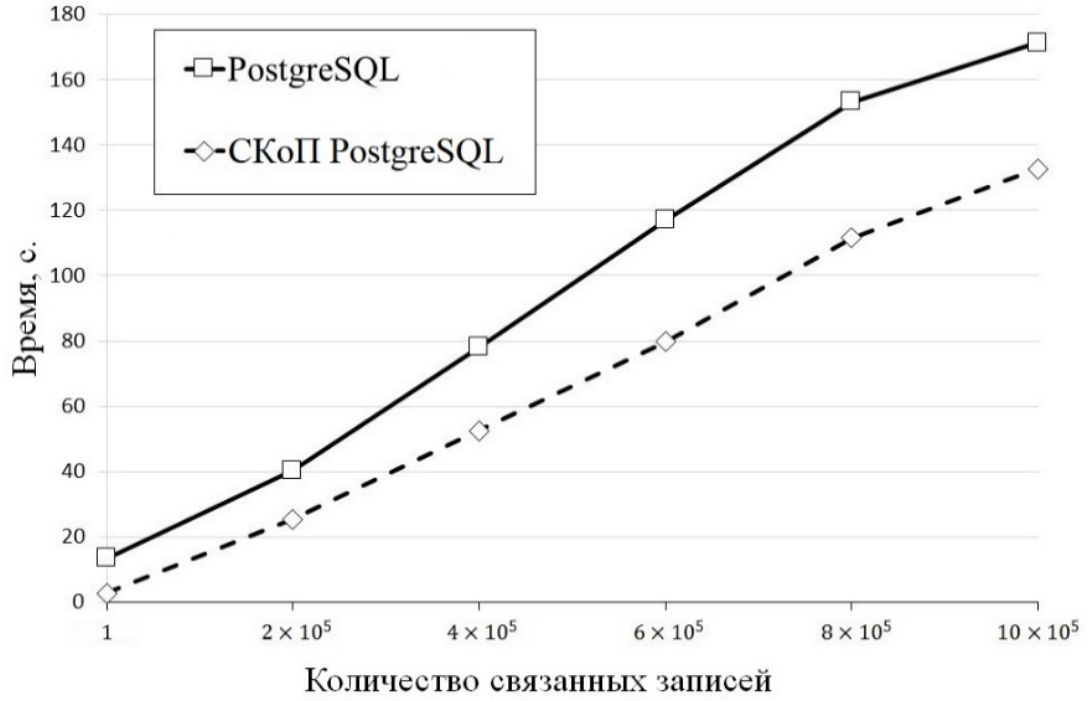

Рис. 11 Эффективность Сопроцессора в СУБД PostgreSQL

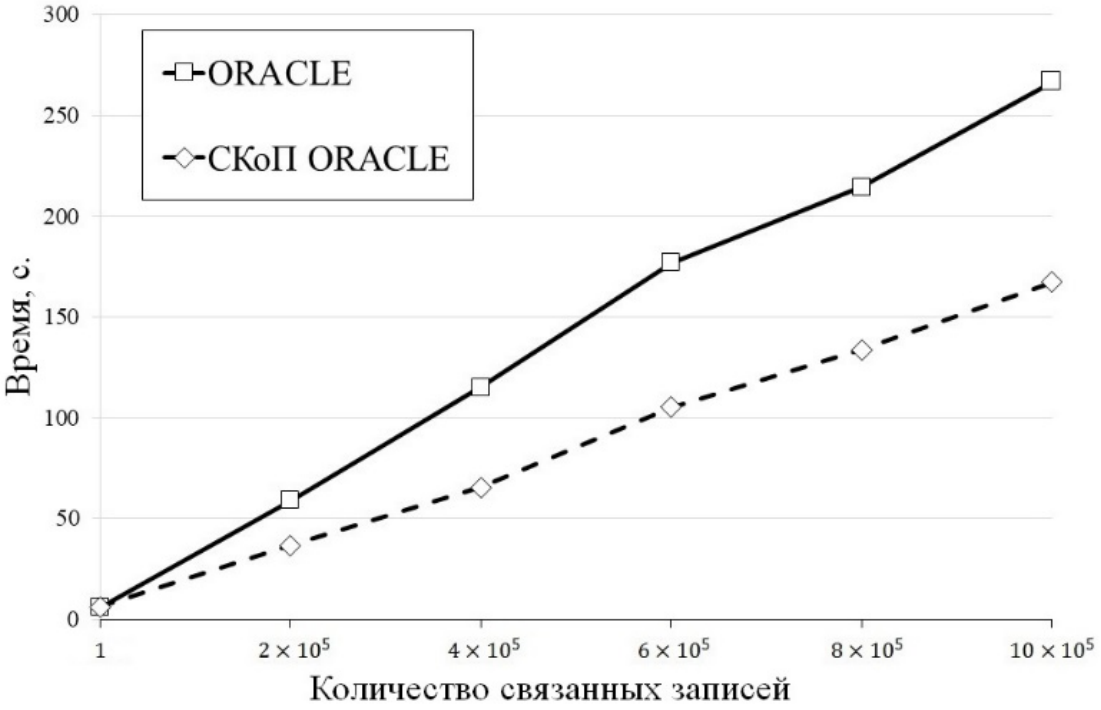

Рис. 12 Эффективность Сопроцессора в СУБД ORACLE

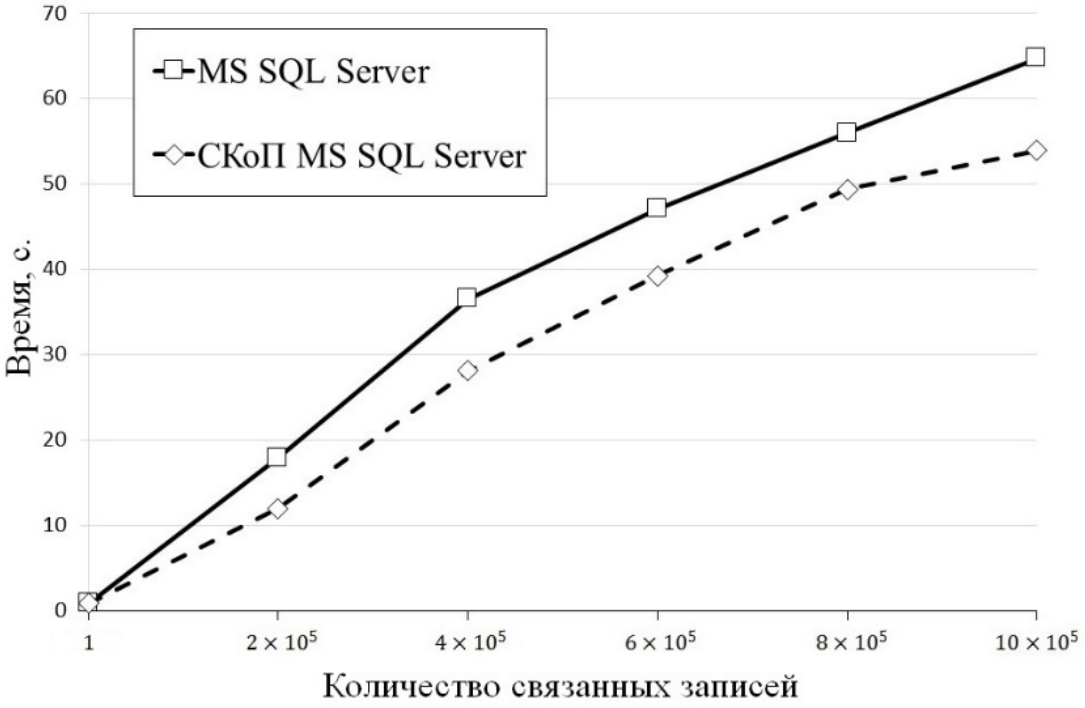

Рис. 13 Эффективность Сопроцессора в СУБД MS SQL Server

4.2.2 Эффективность Сопроцессора в приложениях класса OLTP

В стандартном тесте TPC-E [48] СУБД имитирует торговлю на фондовой бирже и включает в себя следующие три сценария выполнения транзакций:

Транзакция Trade-Update предназначена для эмуляции процесса внесения незначительных исправлений или обновлений в список сделок. Это аналогично процессу, в соответствии с которым клиент или брокер просматривает список сделок и обнаруживают, что требуются некоторые незначительные корректирующие исправления. Различные списки сделок выбираются таким образом, чтобы транзакция формировалась во время выполнения следующих работ:

– обзор общих тенденций рынка;
– обзор сделок за период времени до последнего выписки по счету;
– обзор прошлых показателей конкретной безопасности.

Транзакция Market-Feed предназначена для имитации процесса отслеживания текущей рыночной активности. Это описывает работу брокера, обрабатывающего история сделок с рынка биржи.

Транзакция Trade-Cleanup используется для отмены всех отложенных или отправленных сделок из базы данных.

Применительно к сопроцессору данные транзакции можно описать в виде: обновление информации о сделке (Trade-Update), очистка информации об специфицированной сделке (Trade-Cleanup) и Market-Feed (протоколирование текущей рыночной активности), в каждом из которых выполняется обновление от четырех до шести отношений. Эксперименты проводились с использованием рассмотренного выше сопроцессора pgCOPCT.

Для каждого из указанных сценариев теста TPC-E было сформировано многотабличное представление, обновление которого выливается в выполнение транзакции, которая выполняет обновление отношений БД в соответствии со сценарием. Выполнение каждого сценария с использованием СКоП осуществлялось в режимах холодного и горячего запуска. Холодному запуску СКоП соответствует ситуация, когда Парсер сначала формирует Локальный словарь БД, а затем Коммутатор выполняет необходимые действия. Горячий запуск СКоП означает, что необходимость формирования Локального словаря БД отсутствует и Коммутатор сразу выполняет необходимые действия.

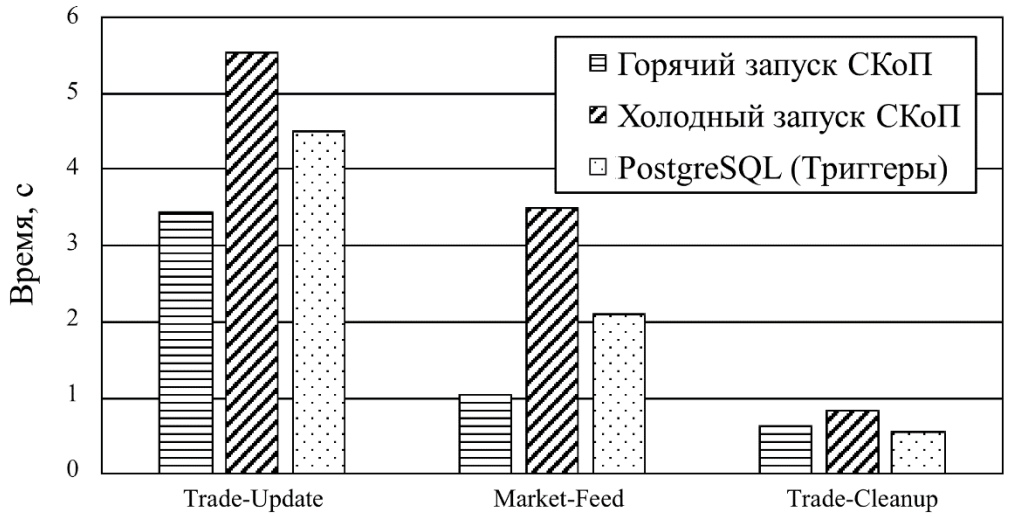

Рис. 14 Эффективность СКоП в приложениях класса OLTP

Результаты экспериментов представлены на Рис. 14. Можно видеть, что для каждого из рассмотренных сценариев в режиме горячего запуска pgCOPCT выполняет обновление многотабличного представления быстрее, чем СУБД PostgreSQL с помощью триггеров. Однако использование триггеров выгоднее в режиме холодного запуска. Таким образом, накладные расходы на поддержку Локального словаря БД являются необходимой платой за эффективность СКоП.

4.3 Выводы

В данной главе рассмотрен подход к формированию и использованию многотабличных приложений на основе запросов проекция-селекция-соединение. В такую форму представления трансформируется большинство пользовательских запросов, сформулированных на естественном языке. Основным преимуществом представленного приложения является расширенные возможности обновления представлений, полученных из реляционной БД. Теоретические результаты рассмотренного подхода предложены в работе [108].

Рассмотренный подход может быть реализован в рамках любой реляционной СУБД, поскольку для организации интерфейса между СУБД и приложением достаточно использования базисных операторов языка SQL. Подход был реализован для свободной СУБД PostgreSQL и зарегистрирован в Федеральной службе по интеллектуальной собственности [100]. Подробное описание реализации и вычислительные эксперименты опубликованы в [104].

ЗАКЛЮЧЕНИЕ

В диссертационной работе рассмотрены методы анализа, построения и использования целостности данных в БД. Рассмотрены условия поддержки целостности реляционной БД на двух уровнях архитектуры ANSI–SPARC. На внешнем уровне для случая поддержки целостности данных при обновлении многотабличных представлений и на концептуальном уровне для поддержки ссылочных ограничений целостности, основанных на новых типизированных зависимостях включения.

На основании введенного оригинального соответствия кортежей была разработана формальная теория зависимостей включения, которая являются основой для построения ссылочной целостности. В рамках этой теории допускается ограниченное использование неопределенных значений. Данное ограничение позволяет сохранить свойство транзитивности для ссылочной целостности, что, в свою очередь, делает формальную теорию естественным обобщением классической теории зависимостей в БД. Рассмотренная теория касается только типизированных зависимостей без использования нетипизированных зависимостей. Такое ограничение согласуется с методологией проектирования реляционных БД: связи на схеме БД не должны обладать какой-либо семантикой, что не выполнено при реализации на схеме нетипизированных зависимостей включения. Кроме того, типизация позволяет избавиться от неоднозначной семантической интерпретации атрибутов отношений, что в конечном итоге гарантирует выполнение логической независимости приложений от схемы БД.

При рассмотрении целостности данных на верхнем уровне архитектуры ANSI–SPARC была разработана теория и подход к обновлению многотабличных представлений. Применение свойства коммутативности гарантирует корректность преобразований. Реализация подхода в среде PostgreSQL позволяет использовать её во многих информационных системах.

Корректность предложенных методов и алгоритмов во всех случаях подтверждена строгими математическими доказательствами.

Основные результаты работы заключаются в следующем.

1. Предложена система аксиом типизированных зависимостей включения с неопределенными значениями в реляционных БД и доказана ее полнота и непротиворечивость.

2. Разработаны алгоритмы построения неизбыточного множества типизированных зависимостей включения с обоснованием их корректности и оценкой вычислительной сложности.

3. Сформулированы и доказаны теоремы о коммутативных преобразованиях для обновления многотабличных представлений БД. Разработана архитектура сопроцессора коммутативных преобразований (СКоП) СУБД и реализован СКоП СУБД PostgreSQL.

4. Проведены вычислительные эксперименты, подтверждающие эффективность предложенных подходов.

Основные результаты диссертационного исследования являются новыми и отсутствуют в работах других авторов, обзор которых был представлен в главе 1. Рассмотрим принципиальные отличия.

Практически во всех работах, посвященных исследованию зависимостей включения, рассматриваются нетипизированные зависимости включения. Как следствие, специфика таких зависимостей определяет содержание таких исследований, например, перестановки атрибутов и взаимодействие с функциональными зависимостями. Для типизированных зависимостей включения эти проблемы не актуальны. Однако, принципиальное отличие данной работы, которая была развита в [83, 84, 94, 97, 103, 107, 109], в том, что вводится условие предшествования на кортежах, в котором неопределенное значение атрибута в подчиненном отношении может соответствовать

определенному или неопределенному значению атрибута в главном отношении. Тогда как, неопределенному значению атрибута в главном отношении может соответствовать только неопределенное значение в подчиненном отношении. Такое условие сохраняет свойство транзитивности для зависимостей с неопределенными значениями и делает формальную теорию естественным обобщением классической теории зависимостей. Нарушение последнего ограничения приводит к смене системы аксиом и всех последующих результатов, например [38]. Следует отметить оригинальность доказательство полноты системы аксиом, представленное в данной работе. Все без исключения исследователи зависимостей включения ссылаются на доказательство полноты в работе [9]. Однако там представлена схема доказательства, относительно которой надо доказать, что логическое следствие зависимостей имеет место для любой последовательности, а не только сгенерированной по предложенному правилу.

Автоматизация поиска зависимостей включения и их реализация на схеме БД с использованием связей является актуальной и довольно старой проблемой. Большинство работ по этой тематике посвящено исследованию алгоритмов поиска зависимостей в заполненной БД. Однако, зависимости включения задают ссылочные ограничения целостности, которые игнорируются при заполнении БД и, следовательно, могут быть нарушены. Это затруднит последующий поиск зависимостей в заполненной БД. В данной работе предлагается предварительная установка на пустой БД ссылочных ограничений целостности, основанных на первичных ключах отношений [81, 82, 93, 95, 96, 99, 102], что сократит количество ошибок при заполнении БД. Изначально предполагалась реализация методов, представленных в работе [105] и расширение возможности раннего обнаружения ссылочных ограничений целостности. Однако, анализ показал о наличии ограничений у метода, что привело к необходимости его существенной модификации.

Кроме того, последующие исследования не выявили какой-либо дополнительной взаимосвязи между ограничениями целостности на схеме БД.

Проблеме обновления многотабличных представлений посвящено значительное количество работ различных авторов. Использование формы реляционного запроса «проекция-селекция-соединение» и ограничение семантики приложения одним целевым отношением БД позволили получить формально строгое (в терминах реляционной алгебры) и корректное (коммутативное) преобразование данных, доказательство которого было опубликовано в [108]. В результате чего было получено универсальное решения для обновления представлений, которое расширяет возможности СУБД [30], путем снятия ограничения соответствия обновляемой строки в представлении и одного кортежа в хранимой таблице БД.

Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

ЛИТЕРАТУРА

1. *Abelló A., Samos J., Saltor F.* Understanding Facts in a Multidimensional Object-Oriented Model // DOLAP / под ред. J. Hammer. ACM, 2001. P. 32–39.
2. *Atwood T.* An Object-Oriented DBMS for Design Support Applications. Proceedings of the IEEE COMPINT 85, 1985. P. 299–307;
3. *Baklarz G.* DB2 9 for Linux, UNIX, and Windows: DBA Guide 6 edition. NY, USA: IBM Press, 2007. 1136 p.
4. *Bancilhon F., Spyratos N.* Update semantics of relational views // ACM Trans. Database Syst. 1981. Т. 6, № 4. С. 557–575.
5. *Bauckmann J., Abedjan Z., Leser U., Müller H., Naumann F.* Discovering conditional inclusion dependencies // 21st ACM international conference on Information and knowledge management (CIKM '12) (ACM, New York, NY, USA), 2012. С. 2094–2098.
6. *Beeri C., Fagin R., Maier D., Yannakakis M.* On the Desirability of Acyclic Database Schemes // ACM. Т. 38, № 3, 1983. С. 479–513.
7. *Bertossi L., Salimi B.* Causes for Query Answers from Databases: Datalog Abduction, View-updates, and Integrity Constraints // Int. J. Approx. Reason. 2017. Vol. 90. P. 226–252. DOI: 10.1016/j.ijar.2017.07.010.
8. *Biskup J., Dublish P.* Objects in Relational Database Schemes with Functional, Inclusion and Exclusion Dependencies // Theoretical Informatics and Applications. T. 27, 1993. 183–219.
9. *Casanova M., Fagin R., Papadimitriou C.* Inclusion Dependencies and Their Interaction with Functional Dependencies // Journal of Computer and System Sciences. 1984. № 28(1). P. 29-59.
10. *Castro G.F., Pérez I., Piñero P., Torres S., Vásquez M., Hidalgo J., Vera-Lucio N., Valencia-García R., Lagos-Ortiz K., Alcaraz-Mármol G., Cioppo J., Vera-Lucio N.* Platform for Project Evaluation Based on Soft-Computing

Techniques // In book: Communications in Computer and Information Science, vol 658. Springer, Cham. 2016. P. 226–240. DOI: 10.1007/978-3-319-48024-4_18

11. *Chacko A.M., Basheer A. M., Kumar S. D. M.* Capturing provenance for big data analytics done using SQL interface // IEEE UP Section Conference on Electrical Computer and Electronics (UPCON), Allahabad, 2015. P. 1–6. DOI: 10.1109/UPCON.2015.7456749
12. *Chandra A.K., Vardi M.Y.* The Implication Problem for Functional and Inclusion Dependencies is Undecidable // SIAM Journal on Computing. T. 14, № 3, 1985. C. 671–677.
13. *Chaudhuri S., Dayal U.* Data Warehousing and OLAP for Decision Support (Tutorial) // SIGMOD Conference / под ред. J. Peckham. ACM Press, 1997. C. 507–508.
14. *Colliat G.* OLAP, Relational, and Multidimensional Database Systems // SIGMOD Rec. New York, NY, USA, 1996. T. 25, № 3. C. 64–69. DOI: 10.1145/234889.234901.
15. *Cosmadakis S., Kanellakis P., Vardi M.* Polynomial-time implication problems for unary inclusion dependencies // J. ACM. T 37, № 1, 1990. C. 15–46.
16. *Cosmadakis S., Papadimitriou C.* Updates of Relational Views // J. ACM. 1984. Vol. 31, No 4. P. 742–760. DOI: 10.1145/1634.1887.
17. *Dayal U., Bernstein P.A.* On the correct translation of update operations on relational views // ACM Trans. Database Syst. 1982. T. 7, № 3. C. 381–416.
18. *Du J., Meehan J., Tatbul N., Zdonik S.* Towards Dynamic Data Placement for Polystore Ingestion // In Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics (BIRTE '17), Damianos Chatziantoniou, Malu Castellanos, and Panos K. Chrysanthis (Eds.). ACM, New York, NY, USA, 2017. Article 2. P. 1–8. DOI: 10.1145/3129292.3129297

19. *Eisenberg, A., Melton, J.* SQL. ACM SIGMOD Record 1999, 28(1), 131–138. doi:10.1145/309844.310075
20. *Fagin R., Vardi M.Y.* Armstrong databases for functional and inclusion dependencies // Information Processing Letters. T. 16, № 1, 1983. C. 13–19.
21. *Forta B.* SQL in 10 Minutes, Sams Teach Yourself, 4th Edition. USA: Sams Publishing PTG, 2013. 288 c.
22. *Garmany J., Walker J., Clark T.* Logical Database Design Principles. NY: CRC Press, Auerbach Publications, 2005. 69 c.
23. *Ghandeharizadeh S., Yap J.* SQL Query to Trigger Translation: A Novel Transparent Consistency Technique for Cache Augmented SQL Systems // Proceedings of the 28th International Workshop on Database and Expert Systems Applications, DEXA 2017, August 28–31, 2017, Lyon, France. P. 37–41. DOI: 10.1109/DEXA.2017.24.
24. *Gómez-López F.T., Gasca R.M., Pérez-Álvarez J.M.* Compliance validation and diagnosis of business data constraints in business processes // Information Systems. T. 48, 2015. C. 26–43.
25. *Hartmann S., Link S.* The implication problem of data dependencies over SQL table definitions: axiomatic, algorithmic and logical characterizations // ACM Transactions on Database Systems. 2012. T. 37, № 2. C. 1–40.
26. *Hayamizu Y., Kawamichi R., Goda K., Kitsuregawa M.* Benchmarking and Performance Analysis of Event Sequence Queries on Relational Database. TPCTC. 2018. pp. 110–125. DOI: 10.1007/978-3-030-11404-6_9.
27. *Hillyer M.* Sakila Sample Database. URL: https://dev.mysql.com/doc/sakila/en/ (дата обращения: 29.10.2019).
28. *Hsu T., Chang D., Lee H.* The Study of Application and Evaluation with NoSQL Databases in Cloud Computing // International Conference on Trustworthy Systems and their Applications, Taichung. 2014. P. 57–62. DOI: 10.1109/TSA.2014.18

29. *Huang J., Mozafari B., Schoenebeck G., Wenisch T.F.* A Top-Down Approach to Achieving Performance Predictability in Database Systems // In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17). ACM, New York, NY, USA. 2017. P. 745–758. DOI: 10.1145/3035918.3064016
30. Interim Report: ANSI/X3/SPARC Study Group on Data Base Management Systems. FDT - Bulletin of ACM SIGMOD. 1975. Vol. 7, No. 2. P. 1–140.
31. ISO/IEC 9075-11:2016 Information technology. Database languages. SQL. Part 11: Information and Definition Schemas (SQL/Schemata). Washington. 2016. 327 p.
32. *Jamkhedkar P., Johnson T., Kanza Y., Shaikh A., Shankaranarayanan N. K., Shkapenyuk V.* A Graph Database for a Virtualized Network Infrastructure // In Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). ACM, New York, NY, USA. 2018. P. 1393–1405. DOI: https://doi.org/10.1145/3183713.3190653
33. *Jhummarwala A., Alkathiri M., Karamta M., Potdar M. B.* Comparative Evaluation of Various Indexing Techniques of Geospatial Vector Data for Processing in Distributed Computing Environment // In Proceedings of the 9th Annual ACM India Conference (COMPUTE '16). ACM, New York, NY, USA. 2016. P 167–172. DOI: 10.1145/2998476.2998493
34. *Jiménez R. E. L.* Pentesting on web applications using ethical hacking // 2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI), San Jose. 2016. P. 1–6. DOI: 10.1109/CONCAPAN.2016.7942364
35. *Johnson D.S., Klug A.* Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies // Computer and System Sciences. T. 28, 1984. C. 167–189.

36. *Kanellakis P.C., Cosmadakis S.S., Vardi M.Y.* Unary inclusion dependencies have polynomial time inference problems // Proceedings of the fifteenth annual ACM symposium on Theory of computing (STOC '83) (New York, USA), 1983. C. 264–277.

37. *Keller A.* Algorithms for Translating View Updates to Database Updates for Views Involving Selections, Projections and Joins. Proceedings of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS’85, March 25–27, 1985, Portland, USA. ACM, 1985. pp. 154–163. DOI: 10.1145/325405.325423

38. *Köhler H., Link S.* Inclusion dependencies and their interaction with functional dependencies in SQL // Journal of Computer and System Sciences. T. 85, 2017. C. 104–131. DOI: 10.1016/j.jcss.2016.11.004

39. *Köhler H., Link S.* Inclusion Dependencies Reloaded // Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15). ACM, New York, NY, USA, 2015. C. 1361–1370. DOI: 10.1145/2806416.2806539

40. *Kovács F., Dávid Z.* Visual Modeler for Data Intensive Tasks // In Proceedings of the 16th European Conference on Software Maintenance and Reengineering, (CSMR) Szeged, Hungary, 2012, P. 509-512. DOI: 10.1109/CSMR.2012.66

41. *Langerak R.* View updates in relational databases with an independent scheme, ACM Trans. Database Syst. 1990. vol. 15, no 1. pp. 40–66. DOI: 10.1145/77643.77645.

42. *Lechtenbörger J.* The Impact of the Constant Complement Approach Towards View Updating // Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS’03, June 9–11, 2003, San Diego, CA, USA. ACM, 2003. P. 49–55. DOI: 10.1145/773153.773159.

43. *Leighton B., Cox S.J.D., Car N.J., Stenson M.P., Vleeshouwer J., Hodge J.* A Best of Both Worlds Approach to Complex, Efficient, Time Series Data Delivery // In book: IFIP Advances in Information and Communication Technology, vol 448. Springer, Cham. 2015. P. 371–379. DOI: 10.1007/978-3-319-15994-2_37

44. *Levene M., Loizou G.* How to Prevent Interaction of Functional and Inclusion Dependencies // Inf. Process. Lett., T. 71, № 3-4, 1999. С. 115–125. DOI: 10.1016/S0020-0190(99)00095-2

45. *Levene M., Loizou G.* NULL Inclusion Dependencies in Relational Databases // Information and Computation, T. 136, № 2, 1997. С. 67–108. DOI: 10.1006/inco.1997.2631

46. *Levene M., Loizou G.* The Additivity Problem for Data Dependencies in Incomplete Relational Databases / Chapter in «Semantics in Databases» // Lecture Notes in Computer Science, T. 1358, 1998. С. 136–169.

47. *Levene M., Vincent M.W.* Justification for Inclusion Dependency Normal Form // IEEE Trans. on Knowl. and Data Eng. T 12, № 2, 2000. С. 281–291. DOI: 10.1109/69.842267

48. *Li Y., Levine C.* Extending TPC-E to Measure Availability in Database Systems. TPCTC 2011. pp. 111–122. DOI: 10.1007/978-3-642-32627-1_8

49. *Link S., Memari M.* Static Analysis of Partial Referential Integrity for Better Quality SQL Data // Americas Conference on Information Systems, Chicago, USA. 2013. С. 1–10.

50. *Liu W., Yan H., Zhou W., Lei Z.* Network user dial-up behavior analysis // International Conference on Future Information Technology and Management Engineering, Changzhou, 2010. P. 39–43. DOI: 10.1109/FITME.2010.5655804

51. *Lopes S., Petit J.M., Toumani F.* Discovering interesting inclusion dependencies: application to logical database tuning // Information Systems. T. 27, № 1, 2002. С. 1–19.

52. *Lykhenko O., Frolova A., Obolenska M.* Designing the database for microarray experiments metadata // IEEE International Young Scientists Forum on Applied Physics and Engineering (YSF), Lviv. 2017. P. 127–131. DOI: 10.1109/YSF.2017.8126658

53. *Ma S, Fan W, Bravo L.* Extending inclusion dependencies with conditions // Theoretical Computer Science. T. 515, 2014. C. 64–95.

54. *Marchi F.D., Lopes S., Petit J.M.* Efficient Algorithms for Mining Inclusion Dependencies // Advances in Database Technology - EDBT 2002 (Prague, Czech Republic), 2002. C. 199–214.

55. *Masunaga Y.* A Relational Database View Update Translation Mechanism // In Proceedings of the 10th International Conference on Very Large Data Bases (VLDB '84), San Francisco, CA, USA. 1984. C. 309–320.

56. *Masunaga Y., Nagata Y., Ishii T. Y.* Extending the View Updatability of Relational Databases from Set Semantics to Bag Semantics and Its Implementation on PostgreSQL. IMCOM’18, DOI: 10.1145/3164541.3164584.

57. *Masunaga Y., Nagata Y., Ishii T.* Making Join Views Updatable on Relational Database Systems in Theory and in Practice. IMCOM 2019. vol. 935. pp. 823-840, DOI: 10.1007/978-3-030-19063-7_66

58. *Missaoui R., Godin R.* The Implication Problem for Inclusion Dependencies: A Graph Approach // SIGMOD Record. T. 19, № 1, 1990. C. 36–40.

59. *Mosin S.V., Zykin S.V.* Truth Space Method for Caching Database Queries // Моделирование и анализ информационных систем. 2015. Т. 22, № 2. С. 248–258.

60. *Motl J., Kordík P.* Foreign Key Constraint Identification in Relational Databases // Proceedings of the 17th Conference on Information Technologies - Applications and Theory (ITAT 2017), Martinské hole, Slovakia, September 22-26, 2017. C. 106–111.

61. *Nguyen D., Aref M., Bravenboer M., Kollias G., Ngo H.Q., Ré C., Rudra A.* Join Processing for Graph Patterns: An Old Dog with New Tricks // In Proceedings of the GRADES'15 (GRADES'15). ACM, New York, NY, USA, Article 2. 2015. H 1–8. DOI: 10.1145/2764947.2764948

62. Oracle Database Online Documentation 11g Release 1 (11.1). Database Administrator's Guide, 2017. 299 c. URL: https://docs.oracle.com/cd/B28359_01/server.111/b28310/toc.htm (дата обращения: 31.10.17).

63. *Ordonez C., García-García J.* Referential integrity quality metrics // Decis. Support Syst. T. 44, № 2, 2008. C. 495–508. DOI: 10.1016/j.dss.2007.06.004

64. *Pedersen T.B., Jensen C.S., Dyreson C.E.* A Foundation for Capturing and Querying Complex Multidimensional Data // Inf. Syst. Oxford, UK, UK, 2001. T. 26, № 5. C. 383–423. DOI: 10.1016/S0306-4379(01)00023-0.

65. *Pippal S., Singh S., Sachan R.K., Kushwaha D.S.* High availability of databases for cloud // 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2015. P. 1716–1722.

66. *Pokorný J., Valenta M., Kovačič J.* Integrity constraints in graph databases // Procedia Computer Science. T. 109, 2017. C. 975–981. DOI: 10.1016/j.procs.2017.05.456

67. *Shaabani N., Meinel C.* Incremental Discovery of Inclusion Dependencies // Proceedings of the 29th International Conference on Scientific and Statistical Database Management (SSDBM '17). ACM, New York, NY, USA, 2017. Article 2, 12 pages. DOI: 10.1145/3085504.3085506

68. *Sosnin P.* Question-Answer Shell for Personal Expert System. Chapter in the book “Expert Systems for Human, Materials and Automation.” Published by Intech, 2011. C. 51–74.

69. *Sultana S., Dixit S.* Indexes in PostgreSQL // International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, 2017. P. 512–515. DOI: 10.1109/ICIMIA.2017.7975511

70. *Trummer I., Koch C.* Approximation schemes for many-objective query optimization // In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). ACM, New York, NY, USA. 2014. P. 1299–1310. DOI: 10.1145/2588555.2610527

71. *Trummer I., Koch C.* Solving the Join Ordering Problem via Mixed Integer Linear Programming // In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17). ACM, New York, NY, USA. 2017. P. 1025–1040. DOI: 10.1145/3035918.3064039

72. *Tu S., Kaashoek M.F., Madden S., Zeldovich N.* Processing analytical queries over encrypted data. Proc. VLDB Endow. Vol. 6, No 5 (March). 2013. P. 289–300. DOI: 10.14778/2535573.2488336

73. *Verma A., Kumar S., Dixit S.* Data synchronization in heterogeneous database environment // 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida. 2016. P. 536–541. DOI: 10.1109/IC3I.2016.7918022

74. *Visser J.* Coupled Transformation of Schemas, Documents, Queries, and Constraints // Electronic Notes in Theoretical Computer Science. T. 200, № 3, 2008. С. 3–23.

75. *Wang X., Zeldovich N., Kaashoek M.F., Solar-Lezama A.A* Differential Approach to Undefined Behavior Detection // ACM Trans. Comput. Syst. 33, 1, Article 1 (March 2015). P. 1–29. DOI: 10.1145/2699678

76. *Wnuk P., Syfert M.* Efficiency Analysis of Relational and Nonrelational Databases in Application to Archiving Measurements / In book: Advanced Solutions in Diagnostics and Fault Tolerant Control // Springer, Cham. 2018. P. 458–470. DOI 10.1007/978-3-319-64474-5_39

77. *Zhang C., Wang X., Peng Z.* Extracting Dimensions for OLAP on Multidimensional Text Databases // WISM (2). Т. 6988 / под ред. Z. Gong [и др.]. Springer, 2011. С. 272–281. (Lecture Notes in Computer Science).
78. *Zykin S., Zykin V.* Updates of View in Relational Databases // Proceedings of the 12th International Scientific and Technical Conference “Dynamics of Systems, Mechanisms and Machines”, Dynamics 2018, Omsk, Russia, 13-15 November 2018. Article no. 8601495.
79. *Zykin S.V.* Generation of User View for a Relational Database by Mappings // Programming and Computer Software. 1999. Т. 25, № 3. С. 173–183.
80. *Zykin V., Zykin S.* Analysis of Typed Inclusion Dependences with Null Values // Automatic Control and Computer Sciences. 2018. Vol. 52, Iss. 7, P. 638–646.
81. *Zykin V.S.* Automatization of Foreign Keys Construction // «2016 Dynamics of Systems, Mechanisms and Machines (Dynamics)». Omsk, Russia, 2016. P. 1–4. DOI: 10.1109/Dynamics.2016.7819118
82. *Zykin V.S.* Automatization of foreign keys construction // Applied mathematics and fundamental informatics: collection of scientific articles. Publishing house of OmSTU. 2014. P. 65–72.
83. *Zykin V.S.* Integrity constraint of incomplete data // Optimization and applications : Proceedings include extended abstracts of reports presented at the VI International Conference on Optimization Methods and Applications. Moscow, 2015. P. 193.
84. *Zykin V.S.* Theory for Typed Inclusion Dependencies with Null Values // Book of Abstracts of the 2nd International Conference and Summer School «Numerical Computations: Theory and Algorithms». Pizzo Calabro, Italy, 2016. P. 167.
85. *Агафонова М. С., Козярская Л. К.* Совершенствование информационного обеспечения управления организацией // Научно-методический электронный журнал «Концепт». 2017. Т. 39. С. 216–220.

86. *Беляева М.А., Бурляева О.К., Сырова В.И.* Формирование мультимодельной системы для принятия оптимальных управленческих решений на предприятии // Программные продукты и системы. 2014. № 2 (106). С. 181–187.

87. *Горбачев С.И., Булычев С.Н.* Повышение эффективности обучения информационным технологиям при подготовке инженеров-экологов // Интернет-журнал «Мир науки», 2017, Т. 5, № 5. С. 1–8.

88. ГОСТ 20886-85 Организация данных в системах обработки данных. Термины и определения

89. Грасиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. М.: Вильяме, 2003. 1088 с.

90. *Евдошенко О.И., Кравец А.Г., Петрова И. Ю.* Разработка онтологии и базы данных для эффективного поиска научно-технической документации // Прикладная информатика. 2015. Т. 10, № 5(59). С. 85–92.

91. *Ершов А.П., Ильин В.П.* Пакеты программ, как методология решения прикладных задач // Пакеты прикладных программ. М.: Наука. 1982. С. 4–18.

92. *Жижченко А.Б., Изаак А.Д.* Информационная система Math-Net.Ru. Современное состояние и перспективы развития. Импакт-факторы российских математических журналов // Успехи матем. наук. 2009. Т. 64, № 4 (388). С. 195–204.

93. *Зыкин В.С.* Автоматизация формирования внешних ключей // Прикладная математика и фундаментальная информатика: сб. материалов III Рос. молодеж. научн.-практ. конф.: Омск, 2014. № 1. С. 190–193.

94. *Зыкин В.С.* Анализ типизированных зависимостей включения с неопределенными значениями // Прикладная математика и фундаментальная информатика: сб. материалов III Рос. молодеж. научн.-практ. конф.: Омск, 2016. № 3. С. 195–199.

95. *Зыкин В.С.* Инструментальная среда формирования внешних ключей на схеме реляционной базы данных // Омский научный вестник. 2017. № 1(151). С. 140–143.

96. *Зыкин В.С.* Ограничения целостности для неопределенных значений кортежей // Прикладная математика и фундаментальная информатика: сб. материалов III Рос. молодеж. научн.-практ. конф.: Омск, 2015. № 2. С. 227–231.

97. *Зыкин В.С.* Построение минимального покрытия зависимости включения // Тезисы докладов XVI Байкальской международной школы-семинара «Методы оптимизации и их приложения». Иркутск: ИСЭМ СО РАН. 2014. С. 52.

98. *Зыкин В.С.* Программа для построения неизбыточного набора связей на схеме баз данных: свидетельство о государственной регистрации программ для ЭВМ. № 2018661248 от 04.09.2018

99. *Зыкин В.С.* Разработка инструментария для реализации ограничений целостности на данные в корпоративных базах данных // Прикладная математика и фундаментальная информатика: сб. материалов III Рос. молодеж. научн.-практ. конф.: Омск, 2013. С. 91–95.

100. *Зыкин В.С.* Редактор многотабличного представления данных: свидетельство о государственной регистрации программ для ЭВМ. № 2018661249 от 04.09.2018

101. *Зыкин В.С.* Сравнительный анализ различных СУБД при редактировании многотабличных представлений данных // Информационный бюллетень Омского научно-образовательного центра ОмГТУ и ИМ СО РАН в области математики и информатики. 2017. № 1(1). С. 153–154.

102. *Зыкин В.С.* Ссылочная целостность данных в корпоративных информационных системах // Информатика и ее применения. Т. 9, № 3, 2015. С. 119–127. DOI: 10.14357/19922264150310

103. *Зыкин В.С., Зыкин С.В.* Анализ типизированных зависимостей включения с неопределенными значениями // Моделирование и анализ информационных систем. 2017. Т. 24, № 2. Р. 155–167. DOI: 10.18255/1818-1015-2017-2-155-167

104. *Зыкин В.С., Цымблер М.Л.* Обновление многотабличных представлений на основе коммутативных преобразований базы данных // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 2. С. 92–106. DOI: 10.14529/cmse190206

105. *Зыкин С.В*. Автоматизация формирования связей на схеме базы данных // Омский научный вестник. 2010. № 4(58). С. 150–155.

106. *Зыкин С.В.* Области определения функциональных зависимостей в базах данных // Труды Института математики и механики УрО РАН, Т. 23, № 3, 2016. С. 117–129. DOI: 10.21538/0134-4889-2016-22-3-117-129

107. *Зыкин С.В., Зыкин В.С.* Аксиоматика типизированных зависимостей включения с неопределенными значениями в базах данных // Тезисы докладов международной конференции "Математика в современном мире", посвященной 60-летию Института математики им. С.Л. Соболева, Новосибирск, 14-19 августа 2017. С. 78.

108. *Зыкин С.В., Зыкин В.С.* Коммутативные преобразования в базе данных при редактировании многотабличных запросов // Информационные технологии. 2018. Т. 24. № 5. С. 330–338. DOI: 10.17587/it.24.330-338

109. *Зыкин С.В., Зыкин В.С.* Основы теории ограничений целостности в базах данных // В книге: Информационный бюллетень Омского научно-образовательного центра ОмГТУ и ИМ СО РАН в области математики и информатики Материалы VIII Международной молодежной научно-практической конференции с элементами научной школы. 2018. С. 20.

110. *Зыкин С.В., Полуянов А.Н.* Итерирование отношений базы данных для запросов специального вида // Материалы X Международной IEEE

научно-технической конференции «Динамика систем, механизмов и машин», Омск, 15–17 ноября, 2016. Т. 4. № 1. С. 18–21.

111. *Калиниченко Л.А.* Методы и средства интеграции неоднородных баз данных. М.: Наука, 1983. 423 с.
112. *Когаловский М.Р.* Энциклопедия технологий баз данных. М.: Финансы и статистика, 2005. 800 с.
113. *Кузнецов С.Д.* Основы баз данных. 2-е изд. М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. 484 с. ISBN 978-5-94774-736-2.
114. *Мартин Дж.* Организация баз данных в вычислительных системах. М.: Мир, 1980. 664 с.
115. *Мейер Д.* Теория реляционных баз данных. М.: Мир, 1987. 608 с.
116. *Минатуллаев Ш.М., Омарова З.К., Рябов И.М.* Основные принципы повышения эффективности городских перевозок пассажиров и методика конкурсного отбора перевозчиков для ускорения их реализации // Интернет-журнал «Науковедение». 2016. Т. 8, №5. С. 1–9.
117. *Наумов А.Н., Вендров А.М., Иванов В.К.* Системы управления базами данных и знаний. М.: Финансы и статистика, 1991. 352 с.
118. *Ульман Дж.* Основы систем баз данных. М.: Финансы и статистика, 1983. 334 с.
119. *Филиппович А.Ю.* Взаимные функциональные зависимости // Системный администратор. № 1. 2002. С. 84–89.