MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 29, NO. 3, 2022
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Testing Dependencies and Inference Rules in Databases

S. V. Zykin[1]

DOI: 10.18255/1818-1015-2022-3-210-227

[1]Sobolev institute of mathematics SB RAS, 4 Acad. Koptyug av., Novosibirsk 630090, Russia.

MSC2020: 68P15
Research article
Full text in Russian

Received July 30, 2022
After revision August 31, 2022
Accepted September 2, 2022

The process of testing dependencies and inference rules can be used in two ways. First, testing allows verification hypotheses about unknown inference rules. The main goal, in this case, is to search for the relation - a counterexample that illustrates the feasibility of the initial dependencies and contradicts the consequence. The found counterexample refutes the hypothesis, the absence of a counterexample allows searching for a generalization of the rule and conditions for its feasibility (logically imply). Testing cannot be used as a proof of the feasibility of inference rules, since the process of generalization requires the search for universal inference conditions for each rule, which cannot be programmed, since even the form of these conditions is unknown. Secondly, when designing a particular database, it may be necessary to test the feasibility of a rule for which there is no theoretical justification. Such a situation can take place in the presence of anomalies in the superkey. The solution to this problem is based on using join dependency inference rules. For these dependencies, a complete system of rules (axioms) has not yet been found. This paper discusses: 1) a technique for testing inference rules using the example of join dependencies, 2) a scheme of a testing algorithm is proposed, 3) some hypotheses are considered for which there are no counterexamples and inference rules, 4) an example of using testing when searching for a correct decomposition of a superkey is proposed.

Keywords: relational databases; join dependencies; inference rules; testing

INFORMATION ABOUT THE AUTHORS

Sergey V. Zykin
correspondence author

orcid.org/0000-0002-0576-2149. E-mail: szykin@mail.ru
leading researcher, Doctor of Technical Sciences.

Funding: The research was funded in accordance with the state task of the IM SB RAS, project FWNF-2022-0016.

For citation: S. V. Zykin, “Testing Dependencies and Inference Rules in Databases”, *Modeling and analysis of information systems*, vol. 29, no. 3, pp. 210-227, 2022.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 29, № 3, 2022
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Тестирование зависимостей и правил вывода в базах данных

С. В. Зыкин[1]

DOI: 10.18255/1818-1015-2022-3-210-227

[1]Институт математики им. С. Л. Соболева СО РАН, пр. ак. Коптюга, д. 4, г. Новосибирск, 630090 Россия.

УДК 004.652.4
Научная статья
Полный текст на русском языке

Получена 30 июля 2022 г.
После доработки 31 августа 2022 г.
Принята к публикации 2 сентября 2022 г.

Процесс тестирования зависимостей и правил вывода может быть использован в двух направлениях. Во-первых, тестирование позволяет проверить гипотезы относительно неизвестных правил вывода. Основная цель при этом поиск реализации отношения – контрпримера, который удовлетворяет исходным зависимостям и противоречит следствию. Найденный контрпример опровергает гипотезу, отсутствие контрпримера позволяет приступить к поиску обобщения правила и условий его выполнимости (logically imply). Тестирование не может служить доказательством выполнимости правил вывода, поскольку процесс обобщения требует поиска универсальных условий выводимости для каждого правила, что невозможно запрограммировать, поскольку даже вид этих условий неизвестен. Во-вторых, при проектировании конкретной базы данных может потребоваться проверка выполнимости правила, для которого отсутствует теоретическое обоснование. Такая ситуация может проявиться при наличии аномалий в суперключе. Решение этой проблемы основывается на использовании правил вывода зависимостей соединения. Для этих зависимостей пока не найдена полная система правил (аксиом). В данной статье рассматривается: 1) методика проведения тестирования правил вывода на примере зависимостей соединения, 2) предложена схема алгоритма тестирования, 3) рассмотрены некоторые гипотезы, для которых отсутствуют контрпримеры и правила вывода, 4) предложен пример использования тестирования при поиске корректной декомпозиции суперключа.

Ключевые слова: реляционные базы данных; зависимости соединения; правила вывода; тестирование

ИНФОРМАЦИЯ ОБ АВТОРАХ

Сергей Владимирович Зыкин
автор для корреспонденции

orcid.org/0000-0002-0576-2149. E-mail: szykin@mail.ru
вед. науч. сотр., доктор техн. наук.

Финансирование: Работа выполнена в рамках государственного задания ИМ СО РАН, проект FWNF-2022-0016.

Для цитирования: S. V. Zykin, “Testing Dependencies and Inference Rules in Databases”, *Modeling and analysis of information systems*, vol. 29, no. 3, pp. 210-227, 2022.

Введение

Проектирование логической структуры базы данных традиционно основывается на зависимостях, которым удовлетворяют данные в прикладной области [1, 2]. Длительный период исследования таких зависимостей показал, что основой при проектировании схемы базы данных (БД) являются функциональные зависимости, многозначные зависимости, зависимости соединения [1—3] и зависимости включения [4—6]. В данной работе рассматриваются вопросы моделирования зависимостей соединения, которые находят свое применение на практике при декомпозиции суперключа и построении пятой нормальной формы.

Пусть задана реляционная база данных: $(R_1, R_2, \dots, R_k)$, полученная в результате синтеза отношений R_i, $[R_i]$ – совокупность атрибутов (схема отношения R_i), $R_i[V]$ – проекция отношения R_i на множество атрибутов V, $U = \{A_1, A_2, \dots, A_n\}$ – конечное множество всех атрибутов, на которых заданы отношения БД.

Классическое определение зависимости соединения представлено в работах [1, 2]:

Определение 1. Зависимость соединения $\bowtie (X_1, X_2, \dots, X_k)$ выполнима (implies), если для любой реализации отношения R выполнено:

$$R[X] = R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k], \tag{1}$$

где $\bowtie$ – операция естественного соединения [1], $X = X_1 \cup X_2 \cup \dots \cup X_k$.

В определении 1 под любой реализацией отношения R понимается таблица, содержимое кортежей которой соответствует данным в прикладной области. Другими словами, данные в прикладной области удовлетворяют зависимости $\bowtie (X_1, X_2, \dots, X_k)$.

В работе [7] используется понятие полной (full) зависимости соединения. Пусть $S = \{X_1, X_2, \dots, X_k\}$ схема БД, удовлетворяющая условию полноты, тогда $\bowtie (X_1, X_2, \dots, X_k)$ называется полной зависимостью соединения. При этом, схема БД S называется полной, если $attr(S) = U$, где $attr(S) = X_1 \cup X_2 \cup \dots \cup X_k$.

В работе [8] рассмотрено понятие области определения зависимости соединения:

Определение 2. Областью определения зависимости соединения $\bowtie (X_1, X_2, \dots, X_k)$ в отношения R будем называть множество $X = X_1 \cup X_2 \cup \dots \cup X_k \subseteq U$.

Таким образом, зависимость $\bowtie (X_1, X_2, \dots, X_k)$ является полной в проекции $R[X]$, где $X = X_1 \cup X_2 \cup \dots \cup X_k$, и в общем случае она является встроенной в отношении R. В данном случае уместно говорить об области определения зависимости. Далее будем рассматривать полные зависимости соединения, но в собственной области определения.

Если множество зависимостей соединения, которым подчиняется отношение R, гарантирует выполнение какой-либо другой зависимости (логическое следствие), то такое свойство будем записывать в виде правила.

Определение 3. Правило $\bowtie (X_1, X_2, \dots, X_k), \dots, \bowtie (Z_1, Z_2, \dots, Z_n) \vDash \bowtie (Y_1, Y_2, \dots, Y_m)$ выполнимо, если для любой реализации отношения R, удовлетворяющей зависимостям соединения $\bowtie (X_1, X_2, \dots, X_k), \dots, \bowtie (Z_1, Z_2, \dots, Z_n)$, имеет место равенство:

$$R[Y] = R[Y_1] \bowtie R[Y_2] \bowtie \dots \bowtie R[Y_m],$$

где $Y = Y_1 \cup Y_2 \cup \dots \cup Y_m$.

Замечание. При выполнении условий определения 3 зависимость $\bowtie (Y_1, Y_2, \dots, Y_m)$ является логическим следствием зависимостей $\bowtie (X_1, X_2, \dots, X_k), \dots, \bowtie (Z_1, Z_2, \dots, Z_n)$. С использованием выполнимых правил формируются правила вывода (аксиомы), которые позволяют построить не избыточное и непротиворечивое множество зависимостей. Такое множество зависимостей, в свою очередь, позволяет сформировать не избыточное и непротиворечивое множество отношений БД.

Для определения выполнимости зависимости $\bowtie (Y_1, Y_2, \ldots, Y_m)$ традиционно используется вычислительный метод, называемый прогонкой (chase) [2]. Исходными данными для метода являются зависимости соединения и функциональные зависимости, которым должна удовлетворять любая реализация отношения R. Для проверки выполнимости произвольной зависимости соединения строится табло (tableau), соответствующее этой зависимости. Затем к кортежам табло применяются преобразования, соответствующие исходным зависимостям. Если в результате преобразований получается кортеж, состоящая из выделенных переменных (a_j), тогда зависимость соединения выполнима, в противном случае – невыполнима. Для выполнимости функциональной зависимости требуется наличие столбца, состоящего из выделенных переменных.

Достоинства и недостатки прогонки неоднократно обсуждались в научной литературе. Остановимся только на одном свойстве метода, которое является важным в этой работе. Рассмотрим пример: проверим выполнимость правила $\bowtie (AB, BC) \models \bowtie (ABD, BCD)$, где A, B, C, D – отдельные атрибуты. Начальное табло для зависимости $\bowtie (ABD, BCD)$ представлено в таблице 1. Везде далее в качестве имен отдельных атрибутов используются прописные начальные символы латинского алфавита, последние символы – в качестве множеств атрибутов.

Table 1. Dependence tableau

Таблица 1. Табло зависимости

A	B	C	D
a_1	a_2	b_1	a_4
b_2	a_2	a_3	a_4

Применение правила $\bowtie (AB, BC)$ для преобразования табло вызывает затруднение, поскольку правило не содержит атрибута D. Можно предположить, что оба кортежа табло соединимы по атрибутам B и D, поскольку они имеют совпадающие значения. По правилам преобразования табло значение b_1 будет отождествлено с a_3, а значение b_2 будет отождествлено с a_1. Таким образом, в табло появится кортеж, состоящий из выделенных переменных: (a_1, a_2, a_3, a_4). Наличие такого кортежа говорит о выполнимости зависимости $\bowtie (ABD, BCD)$ для любой реализации отношения R, если оно удовлетворяет зависимости $\bowtie (AB, BC)$. Следовательно, правило $\bowtie (AB, BC) \models \bowtie (ABD, BCD)$ выполнимо и может быть использовано для формирования правила вывода $\bowtie (AB, BC) \vdash \bowtie (ABD, BCD)$. Однако представленное предположение и его обоснования являются истинными для полных зависимостей соединения [2, 7]. Зависимость в левой части правила не является полной. Анализ этого правила в следующем примере демонстрирует важность этого условия.

Пример 1. С использованием метода, предложенного в данной работе, была получена реализация (контрпример) отношения R (таблица 2). В представленной таблице 2 выполняется равенство

Table 2. Counterexample 1

Таблица 2. Контрпример 1

A	B	C	D
2	1	2	2
1	1	2	1
2	1	2	1
1	1	1	1

$R[ABC] = R[AB] \bowtie R[BC]$, тогда как соединение $R[ABD] \bowtie R[BCD]$ содержит кортеж $(2, 1, 2, 1)$, а проекция $R[ABCD]$ такой кортеж не содержит. Следовательно, правило $\bowtie (AB, BC) \models \bowtie (ABD, BCD)$ невыполнимо и не может использоваться для формирования правила вывода.

Представленный пример 1 является обоснованием выбора метода, предложенного в данной работе: метод может быть использован для анализа как для встроенных, так и для полных зависимостей соединения.

1. Связанные работы

Большинство публикаций по данной тематике ориентированы на исследование зависимостей соединения как таковых. Однако анализу правил вывода на основе зависимостей соединения, за исключением метода прогонки, должного внимания уделено не было. Рассмотрим некоторые работы, полезные с точки зрения моделирования зависимостей соединения, которые дают представление о вычислительной сложности этой проблемы и содержат полезную информацию об организации самого процесса моделирования.

В работе [9] представлена прогонка (chase) – базовый вычислительный метод для проверки выполнимости некоторой зависимости данных σ на произвольной реализации отношения R при условии, что R удовлетворяет множеству зависимостей данных Σ. Для достижения поставленной цели в прогонке используется табло (tableaux). Поочередно интерпретируя табло как представление (instance) или как шаблон для отношений можно проверить выполнимость функциональной зависимости, многозначной зависимости, зависимости соединения.

Проверка выполнимости зависимости σ на отношении R, удовлетворяющего множеству зависимостей Σ (функциональные, многозначные, соединения) исследуется в [10]. Получено, что в общем случае задача является NP-сложной (NP-hard). Также показано, что проблема проверки возможности применения правила для зависимости соединения к табло T является NP-полной (NP-complete). Следовательно, проверка свойства соединения без потерь информации (lossless join property) для множества отношений является NP-полной. Кроме того, доказано, что гарантируется обнаружение контрпримера для зависимости $\bowtie (X_1, X_2, X_3)$, если реализация отношения R содержит не менее $2^k + 1$ кортежей, где k – количество многозначных зависимостей в Σ.

В работе [11] продолжены исследования вычислительной сложности проверки выполнимости зависимости, если выполнимо некоторое множество зависимостей в отношении базы данных. Показано, что NP-сложной задачей является определение выполнимости зависимости соединения, если выполнимо множество многозначных зависимостей с учетом или без учета функциональных зависимостей. Остается открытым вопрос для зависимостей соединения с количеством компонентов больше 2, если их использовать в качестве обобщения многозначных зависимостей.

Исследованию нескольких формальных систем зависимостей, которые генерируют кортежи и равенства, посвящена работа [12]. Показано, что существуют три типа систем, основанные на подстановке, исключении кортежей и транзитивности. При этом, системы исследуются на нескольких подклассах: общие зависимости, зависимости шаблонов и двоичные зависимости. Показано, что поиск формальной системы для встроенных многозначных зависимостей эквивалентен решению проблемы логического следования для этого класса.

В работе [13] продолжено исследование зависимостей, названных зависимостями, генерирующими равенство. Для полных зависимостей прогонка представляет собой процедуру принятия решения с экспоненциальным временем для проблемы логического следования. В некоторых ограниченных случаях проблема допускает модификацию, когда решение достигается за полиномиальное время. Показано несколько случаев, для которых прогонка является процедурой принятия решения.

Исследованию цикличности гиперграфов, соответствующих зависимостям соединения, посвящена работа [14]. В ней показано, как метод декомпозиции может быть использован для уменьшения вычислительных затрат при проверке целостности данных. Для этого предлагается найти разложение, минимизирующее число ребер наибольшего элемента в соответствии с ранее предложенным критерием. Этот критерий минимальности приводит к определению степени цикличности, которая

позволяет классифицировать зависимости соединения по n-цикличности (ацикличность является частным случаем при $n = 2$). Показано, что для фиксированного значения n проверка целостности может быть выполнена за полиномиальное время для n-циклических зависимостей соединения. Доказано, что n-цикличность влечет глобальную непротиворечивость. Таким образом, проверка глобальной согласованности будет решаема за полиномиальное время, если рассматривать только n-циклические зависимости соединения для фиксированного значения n.

В работе [15] представлена система для обнаружения зависимостей данных, основанная на SQL запросах к базе данных. Основные усилия были направлены на обнаружение ограничений домена, унарных зависимостей включения и функциональных зависимостей в реляционных базах данных. В системе реализуется логический вывод для минимизации доступа к базе данных. Преимущество системы перед остальными подходами заключается в способности обрабатывать большие объемы данных и возможность использования для любых СУБД, в которых реализован язык запросов SQL.

Вероятностный подход к построению множества функциональных зависимостей предложен в работе [16]. Для этого используются различные меры для ошибки определения зависимости в таблице базы данных. Ошибка имеет значение 0, если зависимость выполнена, и значение близкое к 1, если зависимость явно не выполняется. Все зависимости с ошибкой не менее некоторой фиксированной величины могут быть обнаружены с высокой вероятностью. Показано, как алгоритм машинного обучения, созданный *Angluin*, может применяться для получения почти оптимального покрытия за полиномиальное время.

Нормальные формы вложенных отношений рассматриваются в работе [17]. Исследуется проблема преобразований между различными иерархическими структурами данных без потери информации. Отмечено, что эта проблема актуальна во многих областях приложений баз данных, таких как обработка представлений, интеграция схемы и эволюция схемы. В работе предложены операторы реструктуризации дерева схем данных, называемых *COMPRESS*, и доказано, что две схемы данных эквивалентны, когда одна схема преобразуется в другую с помощью конечной последовательности операторов *COMPRESS*. С подобной проблемой столкнулись разработчики и исследователи многомерных баз данных. Корректность представления данных достигалась за счет преобразования реляционного представления данных в иерархическое представление. Позднее было предложено обобщение данного подхода [18], в котором корректность достигается за счет зависимостей соединения.

Наиболее близка к рассматриваемой тематике работа [19], в которой используются табло прогонки и аналитическое табло (the Chase and analytical tableau) в качестве инструмента автоматического создания реализаций баз данных, удовлетворяющих значительному количеству ограничений целостности. Прогонка позволяет находить и отклонять избыточные ограничения. Аналитические таблицы позволяют найти все минимальные множества возможных ограничений. Логические и многозначные зависимости являются семантическими ограничениями, которые очень распространены в практике баз данных. Их семантическое следование эквивалентно логическому следованию фрагментов пропозициональных формул. Авторы продемонстрировали, как можно использовать аналитические таблицы для генерации всех отношений минимального размера, которые удовлетворяют всем ограничениям, полученным на данный момент, но являются контрпримером для текущего претендента на ограничение. Отличие исследований в [19] от данной работы в том, что не ставится цель анализа правил вывода, а только определяются избыточные и противоречивые зависимости.

Алгоритмы интеллектуального анализа данных для вывода зависимостей включения в базе данных предложены в статье [20]. При этом предлагается двухэтапный подход. На первом этапе обнаруживаются унарные зависимости включения за счет оригинального алгоритма, для которого предложена эффективная реализация. На втором этапе выполняется вывод n-арных зависимостей

включения. Второй шаг выполняется по аналогии с алгоритмами интеллектуального анализа данных. Для решения проблемы наличия несоответствий в реальных базах предложено рассматривать неточные зависимости включения, которые допускают незначительное количество ошибок.

В работе [21] представлено исследование алгоритмов проверки выполнимости зависимостей соединения с точки зрения количества операций ввода-вывода во внешнюю память компьютера. Доказана NP-сложность проверки: удовлетворяет ли отношение базы данных определенной зависимости соединения, даже если все схемы отношений в зависимости имеют только 2 атрибута. Представлен эффективный с точки зрения ввода-вывода алгоритм проверки существования нетривиальной зависимости соединения, удовлетворяющей текущему состоянию базы данных.

2. Свойства зависимостей и правил, используемые при тестировании

В результате проведенного анализа литературы получено, что проверка выполнимости зависимости соединения является экспоненциальной как по памяти, так и по времени (количеству операций). При этом не имеет значения метод, который используется для проверки. Поскольку в данной работе основной целью является поиск контрпримера, то выбор был сделан в пользу генерации отношений с проверкой выполнимости заданных зависимостей. В этом случае контрпример будет обнаружен раньше, чем произойдет перебор всех возможных вариантов. Тогда как в методе прогонки необходимо сделать все возможные подстановки, чтобы убедиться в невыполнимости проверяемой зависимости. Контрпример для логического следствия $\Sigma \models \sigma$ считается найденным, если текущая реализация R удовлетворяет множеству зависимостей Σ и не удовлетворяет зависимости σ.

Поскольку поиск контрпримера остается экспоненциальной задачей, то актуальной является минимизация количества возможных вариантов реализации отношений. Другими словами, необходимо избегать генерации отношений, эквивалентных уже рассмотренным реализациям, и необходимо избегать избыточных проверок при анализе выполнимости зависимости. В данном разделе рассмотрим эти вопросы.

Генерируемое отношение в общем виде представлено в таблице 3. Символом n в таблице 3

Table 3. General view of the relation **Таблица 3.** Общий вид отношения

$R =$

A_1	A_2	A_3	…	A_n
a_{11}	a_{12}	a_{13}	…	a_{1n}
a_{21}	a_{22}	a_{23}	…	a_{2n}
…	…	…	…	…
a_{m1}	a_{m2}	a_{m3}	…	a_{mn}

обозначена арность отношения R (количество столбцов), m – кардинальность отношения (количество кортежей), A_j – имя атрибута, a_{ij} – значения атрибута A_j. Допустим, что каждое значение a_{ij} является целочисленным в интервале от 1 до l. Тогда общее количество реализаций отношения R равно l^{nm}.

Наибольший вклад в количество генерируемых отношений вносит интервал значений a_{ij}. При проверке выполнимости зависимости соединения для реализации R сравниваются значения одноименных атрибутов. Поскольку требуется только проверка на равенство этих значений, то для генерации различных реализаций R достаточно использовать два значения, например, 1 и 2. Заметим, что такой интервал тестовых значений приемлем, если не учитывать возможное присутствие неопределенных значений $Null$ для некоторых атрибутов в реальных базах данных. Результат сравнения этого значения, в том числе и с другим значением $Null$, в команде SQL получит значение $unknown$, что воспрепятствует склейке кортежей при выполнении операции естественного соединения. Поясним ситуацию на простом примере. Пусть кортеж t принадлежит отношению R, пусть

X и Y произвольные множества атрибутов и A_j произвольный атрибут, для которого выполнено: $A_j \in X \cap Y$ и $t[A_j] = Null$. По построению кортеж $t[X \cup Y]$ принадлежит проекции $R[X \cup Y]$ и, по правилам выполнения операции естественного соединения, кортеж $t[X \cup Y]$ не принадлежит $R[X] \bowtie R[Y]$. Исключение составляет случай, когда кортеж $t[X \cup Y]$ будет сформирован за счет других кортежей проекций $R[X]$ и $R[Y]$. Заметим, что в данном случае перестает быть верной лемма 5.5 [1], поскольку становится возможна ситуация, когда $R[X] \bowtie R[Y] \subset R[X \cup Y]$. Похожая проблема имеет место при тестировании функциональных зависимостей, когда один из атрибутов допускает неопределенное значение. Для тестирования зависимостей при поиске правил вывода в таких условиях рекомендуется расширить интервал значений атрибута еще одним значением, например, 0. В алгоритме тестирования правил вывода должно быть реализовано условие “$0 \neq 0$”.

На следующем этапе необходимо выполнить сокращение количества кортежей в реализации отношения R. Это возможно за счет удаления (игнорирования) дублированных кортежей. Причина не только в необходимости выполнения условий первой нормальной формы (1NF), но и в том, что дублированные кортежи повторяют результат, который получен на оригинальных кортежах. Это верно как для функциональных зависимостей, так и для зависимостей соединения. Если использовать последовательное генерирование всех возможных комбинаций значений a_{ij}, то дублированные кортежи будут присутствовать в значительном количестве на начальных этапах моделирования, когда большинство a_{ij} равны начальному значению, и в конце моделирования, когда большинство a_{ij} равны конечному значению. Кроме того, дублированные кортежи присутствуют в значительном количестве на промежуточном этапе, когда количество атрибутов меньше количества кортежей. Следовательно, схема генерации представлений R должна исключать появление дублированных кортежей, чтобы исключить последующее манипулирование ими при проверке выполнимости всех зависимостей Σ и σ на текущей реализации отношения R.

Сократить количество реализаций R можно за счет использования известных свойств зависимостей, что позволит сократить множество используемых атрибутов. Для функциональных зависимостей такой способ хорошо проработан – построение минимального покрытия [1]. Для зависимостей соединения в работе [8] представлено множество из пяти правил:

P0) $\varnothing \vDash \bowtie (X), X \subseteq U$.

P1) $\bowtie (X_1, X_2, \dots, X_k) \vDash \bowtie (Y_1, Y_2, \dots, Y_m)$, если:

p11) для любого X_i существует Y_j: $X_i \subseteq Y_j$;

p12) для любого Y_j выполнено $Y_j \subseteq X_1 \cup X_2 \cup \dots \cup X_k$.

P2) $\bowtie (X_1, X_2, \dots, X_k) \vDash \bowtie (Y_1, Y_2, \dots, Y_k)$, если:

p21) $X_i \cap (X_1 \cup X_2 \cup \dots \cup X_{i-1} \cup X_{i+1} \dots \cup X_k) \subseteq Y_i$;

p22) $Y_i \subseteq X_i$.

P3) $\bowtie (X_1, X_2, \dots, X_k) \vDash \bowtie (Y_1, Y_2, \dots, Y_k)$, если:

p31) $Y_i = X_i \cup Z$, $Z \subseteq X_1 \cup X_2 \cup \dots \cup X_k$.

P4) $\bowtie (X_1, X_2, \dots, X_k, V)$, $\bowtie (Y_1, Y_2, \dots, Y_l) \vDash \bowtie (X_1, X_2, \dots, X_k, V \cap Y_1, V \cap Y_2, \dots, V \cap Y_l)$, если:

p41) $Y_i \cap Y_j \subseteq V, i \neq j$;

p42) $V \cap (X_1 \cup X_2 \cup \ ldots \cup X_k) \subseteq V \cap (Y_1 \cup Y_2 \cup \dots \cup Y_l)$.

Для каждого правила доказана непротиворечивость (надежность). Кроме того, доказано, что система правил P0–P4 является обобщением правил, предложенных в работах [7] и [22]. Правила P2 и P4 могут непосредственно быть использованы для сокращения количества атрибутов, поскольку в общем случае зависимости в правой части обоих правил имеют меньше атрибутов, чем зависимости в левой части. С использованием представленной системы правил можно получить еще одно полезное свойство, которое позволяет сократить количество компонентов проверяемой зависимости, а значит и количество необходимых проверок. Рассмотрим это свойство.

Определение 4. Зависимость соединения $\bowtie (Y_1, Y_2, \dots, Y_l)$ будем называть редукцией зависимости $\bowtie (X_1, X_2, \dots, X_k)$, если для любого X_i существует Y_j: $X_i \subseteq Y_j$, для любого Y_j существует X_i: $Y_j \subseteq X_i$, и не существует Y_s и Y_p: $Y_s \subseteq Y_p$, $s \neq p$.

Другими словами, произвольная зависимость соединения будет редуцирована, если из нее удалены компоненты, являющиеся подмножеством других компонентов этой зависимости.

Утверждение 1. Произвольная зависимость соединения и ее редукция эквивалентны.

Доказательство. Пусть зависимость $\bowtie (Y_1, Y_2, \dots, Y_l)$ является редукцией зависимости $\bowtie (X_1, X_2, \dots, X_k)$. Необходимо показать, что $\bowtie (X_1, X_2, \dots, X_k) \vDash \bowtie (Y_1, Y_2, \dots, Y_l)$ и $\bowtie (Y_1, Y_2, \dots, Y_l) \vDash \bowtie (X_1, X_2, \dots, X_k)$.

1. Правило $\bowtie (X_1, X_2, \dots, X_k) \vDash \bowtie (Y_1, Y_2, \dots, Y_l)$ выполнимо, поскольку является следствием правила P1: условия p11 и p12 выполнены по определению 4.

2. Правило $\bowtie (Y_1, Y_2, \dots, Y_l) \vDash \bowtie (X_1, X_2, \dots, X_k)$ выполнимо, поскольку, как и предыдущее правило, является следствием правила P1: выполнение условий p11 и p12 гарантируется определением 4. Утверждение 1 доказано.

Замечание 1. Использование редукции вместо исходной зависимости не сократит общее количество реализаций отношения R, однако избавит от необходимости проведения анализа дополнительных проекций, что в итоге сократит общее время проверки выполнимости правила (поиска контрпримера).

Замечание 2. При построении редукции нельзя удалять атрибуты, используемые только в одном X_i, как это делается в алгоритме Грэхема (Graham) [2] при проверке ацикличности гиперграфа. При удалении таких атрибутов будут получены выполнимые зависимости, что следует из правила P2. Однако эти зависимости не будут эквивалентны исходным зависимостям. Для обоснования рассмотрим простейший пример.

Пример 2. Рассмотрим выполнимое правило $\bowtie (AB, BCD) \vDash \bowtie (AB, BC)$, которое является следствием правила P2. Обеим зависимостям в правиле соответствуют ациклические гиперграфы, поскольку применение к ним алгоритма Грэхема дает пустые множества. Однако зависимости не являются эквивалентными. Проверка правила $\bowtie (AB, BC) \vDash \bowtie (AB, BCD)$ показывает наличие контрпримера, представленного в таблице 4. Для этого примера выполняется равенство $R[ABC] = R[AB] \bowtie R[BC]$, тогда как соединение $R[AB] \bowtie R[BCD]$ содержит кортеж (2, 1, 1, 1), а проекция $R[ABCD]$ такой кортеж не содержит. Следовательно, правило $\bowtie (AB, BC) \vDash \bowtie (AB, BCD)$ невыполнимо и представленные в нем зависимости не являются эквивалентными. Заметим, что для поиска контрпримера хватило всего двух кортежей в таблице 4.

Table 4. Counterexample 2 **Таблица 4.** Контрпример 2

$R =$

A	B	C	D
2	1	1	2
1	1	1	1

Далее необходимо подобрать множество атрибутов $\{A_1, A_2, \dots, A_n\}$ так, чтобы, с одной стороны, учесть все детали проверяемых зависимостей, и, с другой стороны, не вводить лишних атрибутов. Предлагаемое решение проще всего продемонстрировать на примере.

Пример 3. Проведем анализ обобщения аксиомы A8 [1] для многозначных зависимостей. Исходный вид аксиомы: если $X \twoheadrightarrow Y$, $V \subseteq Y$, $W \longrightarrow V$ и $W \cap Y = \emptyset$, тогда $X \longrightarrow V$, где $\twoheadrightarrow$ многозначная зависимость и $\longrightarrow$ функциональная зависимость. Обобщением данной аксиомы является правило:

$$\bowtie (XY, X(U \setminus Y)), BC), \bowtie (WV, W(U \setminus V)) \vDash \bowtie (XV, X(U \setminus V)), \tag{2}$$

где U – множество всех атрибутов (аксиома определена только для полных зависимостей). Зависимость $\bowtie (WV, W(U \setminus V))$ не является эквивалентом функциональной зависимости $W \longrightarrow V$, а является

следствием — более слабым ограничением на данные (аксиома A7 [1]). На рисунке 1 представлены все возможные пересечения множеств атрибутов для аксиомы A8. По построению каждому пересечению соответствует минимальное количество атрибутов, но учитывается каждое влияние компонентов зависимостей друг на друга при выполнении операции естественного соединения. На рисунке 1 названия множеств атрибутов из аксиомы A8 изображены полужирным курсивом на

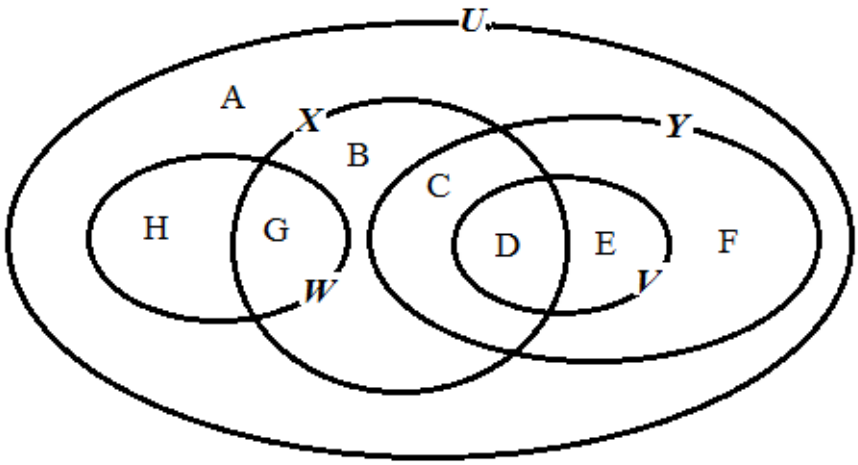

Fig. 1. Permissible Attribute Set Intersections

Рис. 1. Допустимые пересечения множеств атрибутов

границах областей. Каждой области поставлен в соответствие отдельный атрибут. Следовательно, $U = ABCDEFGH$, $X = BCDG$, $Y = CDEF$, $V = DE$ и $W = GH$. Для проверки правила (2) используется следующая модель:

$$\bowtie (BCDEFG, ABCDGH), \bowtie (DEGH, ABCFGH) \models \bowtie (BCDEG, ABCDFG). \tag{3}$$

Правило (2) и его модель (3) не выводимы из правил P0–P4, однако при тестировании формулы (3) контрпример найден не был. Это означает, что можно приступать доказательству выполнимости правила (2).

В работе [2] без доказательства сформулирован признак наличия полной зависимости соединения. Представим формулировку и доказательство этого признака для полных и встроенных зависимостей, чтобы продемонстрировать корректный способ проверки выполнимости и получить количественные характеристики моделируемых объектов.

Утверждение 2. Зависимость $\bowtie (X_1, X_2, \dots, X_k)$ выполнена в R, если для любой реализации R наличие кортежей t_1, t_2,..., t_k, не обязательно различных, гарантирует наличие кортежа $t \in R$: $t[X_i] = t_i[X_i]$, $i = \overline{1,k}$, если $t_i[X_i \cap X_j] = t_j[X_i \cap X_j]$, когда $X_i \cap X_j \neq \emptyset$.

Доказательство. Необходимость. Пусть $X = X_1 \cup X_2 \cup \dots \cup X_k$. Если выполнена зависимость $\bowtie (X_1, X_2, \dots, X_k)$, то для любого отношения R выполнено: $R[X] = R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k]$. Предположим, что отношение R содержит кортежи t_1, t_2,..., t_k, не обязательно различные, для которых $t_i[X_i \cap X_j] = t_j[X_i \cap X_j]$, если $X_i \cap X_j \neq \emptyset$. Тогда кортеж $t' = t_1[X_1] \bowtie t_2[X_2] \bowtie \dots \bowtie t_k[X_k]$ содержится в $R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k]$. Поскольку выполнена зависимость $\bowtie (X_1, X_2, \dots, X_k)$, то кортеж t' принадлежит $R[X]$, следовательно, существует кортеж $t \in R$, $t[X] = t'$ и $t[X_i] = t_i[X_i]$, $i = \overline{1,k}$. Необходимость доказана.

Достаточность. Включение $R[X] \subseteq R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k]$ следует из леммы 5.5 [1]. Покажем, что имеет место обратное включение $R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k] \subseteq R[X]$. Пусть кортеж $t' \in R[X_1] \bowtie R[X_2] \bowtie \dots \bowtie R[X_k]$. Это означает, что в каждой проекции $R[X_i]$, $i = \overline{1,k}$ существует кортеж t_i:

$t_i[X_i] = t'[X_i]$, и $t_i[X_i \cap X_j] = t_j[X_i \cap X_j]$, если $X_i \cap X_j \neq \emptyset$. По условию данного утверждения $t' \in R[X]$. Следовательно, $R[X_1] \bowtie R[X_2] \bowtie \cdots \bowtie R[X_k] \subseteq R[X]$, а значит $R[X] = R[X_1] \bowtie R[X_2] \bowtie \cdots \bowtie R[X_k]$, и зависимость соединения $\bowtie (X_1, X_2, \ldots, X_k)$ выполнима. Утверждение 2 доказано.

Следствие. Если не учитывать значение $Null$, то из доказательства утверждения 2 следует, что при тестировании зависимости $\bowtie (X_1, X_2, \ldots, X_k)$ достаточно проверять условие $R[X_1] \bowtie R[X_2] \bowtie \cdots \bowtie R[X_k] \subseteq R[X]$, то есть каждый кортеж соединения проекций $R[X_i]$ должен принадлежать проекции $R[X]$. Кроме того, для проверки на выполнимость зависимости требуется формировать не менее $k + 1$ различных кортежей, где k количество компонентов в зависимости. В примере 1 максимальное количество компонентов в зависимостях равно двум, однако контрпример для трех кортежей не был найден, для этого потребовалось 4 кортежа. Обусловлено это наличием атрибута в правой части правила, который отсутствует в левой части. С другой стороны, несложно показать невыполнимость правила с таким расположением атрибутов. Условие на количество кортежей не является необходимым при тестировании правила, что демонстрируется в примере 4 и дает возможность обнаружения контрпримера при меньших вычислительных затратах.

Пример 4. Рассмотрим правило, определенное на восьми различных атрибутах, и содержащее по три компонента в правой и левой части:

$$\bowtie (AB, BCDEFG, GH) \vDash \bowtie (ABCE, BEFG, CFGH).$$

Экспериментально установлено, что для этого правила не существует реализации R из двух и менее кортежей, когда зависимости в левой части выполнены, а в правой нет. Проверка правила для трех кортежей показала, что такая реализация R существует, см. таблицу 5.

Table 5. Counterexample 3 **Таблица 5.** Контрпример 3

$R =$

A	B	C	D	E	F	G	H
1	2	1	1	1	2	1	1
1	2	2	1	1	1	1	1
1	1	1	1	1	1	1	1

Поиск контрпримера может занять значительный промежуток времени, если этого контрпримера не существует. В следующем утверждении представлено дополнительное условие прекращения тестирования правила с одной зависимостью в левой части. Пусть $R_X = R[X_1] \bowtie R[X_2] \bowtie \cdots \bowtie R[X_k]$, $X = X_1 \cup X_2 \cup \cdots \cup X_k$, $R_Y = R[Y_1] \bowtie R[Y_2] \bowtie \cdots \bowtie R[Y_l]$ и $Y = Y_1 \cup Y2 \cup \cdots \cup Y_l$.

Утверждение 3. Правило $\bowtie (X_1, X_2, \ldots, X_k) \vDash \bowtie (Y_1, Y_2, \ldots, Y_l)$ выполнимо в R при $X = Y$, если для любой реализации R выполнено: $|R_X| \geq |R_Y|$, где $|R_X|$ – кардинальность отношения R_X.

Доказательство. Для выполнимости правила необходима выполнимость зависимости в левой части этого правила: $\bowtie (X_1, X_2, \ldots, X_k)$. Тогда $|R[X]| = |R_X|$. При $Y = X$ имеем $|R[X]| = |R[Y]|$, следовательно, $|R[Y]| \geq |R_Y|$ и зависимость $\bowtie (Y_1, Y_2, \ldots, Y_l)$ выполнима, что доказывает выполнимость правила $\bowtie (X_1, X_2, \ldots, X_k) \vDash \bowtie (Y_1, Y_2, \ldots, Y_l)$. Утверждение 3 доказано.

Замечание. Если в процессе моделирования по истечении приемлемого времени ни разу не было нарушено условие $|R_X| \geq |R_Y|$, то велика вероятность отсутствия контрпримера и можно приступать к доказательству надежности соответствующего правила вывода. Рассмотренное условие применимо только к правилам с одной зависимостью в левой части.

3. Алгоритм тестирования правил вывода

Ранее была сформулирована цель тестирования – поиск реализации отношения (таблицы) R для правила вывода $\bowtie (X_1, X_2, \ldots, X_k), \ldots, \bowtie (Z_1, Z_2, \ldots, Z_n) \vDash \bowtie (Y_1, Y_2, \ldots, Y_l)$ такого, что зависимости в левой части правила выполнены, а в правой – нет. В этом случае R является контрпримером.

Используя свойства предыдущего раздела, рассмотрим одну из возможных реализаций алгоритма поиска контрпримера. Так как последовательность зависимостей в левой части правила не имеет значения, то на первое место помещаем правило с наибольшим количеством компонентов, чтобы ориентироваться на эту зависимость при выборе количества кортежей в реализации R.

Поскольку организация процесса тестирования довольно громоздкая и не содержит каких-либо структурных сложностей, то ограничимся рассмотрением схемы алгоритма. Входными параметрами являются собственно тестируемое правило и минимальное количество кортежей в реализации $k + 1$, где значение k рекомендуется приравнивать максимальному количеству компонентов в зависимостях тестируемого правила.

Схема алгоритма тестирования правил.

- Шаг 1. Генерация очередной реализации отношения R. Если очередная реализация не существует, то контрпримера нет и конец алгоритма.
- Шаг 2. Дополнение недостающих кортежей из R_X в конец отношения R, где $R_X = R[X_1] \bowtie R[X_2] \bowtie \cdots \bowtie R[X_k]$ и $X = X_1 \cup X_2 \cup \cdots \cup X_k$.
- Шаг 3. Проверка выполнимости оставшихся зависимостей в левой части правила на реализации R. Если существует невыполнимая зависимость, то переход на шаг 1.
- Шаг 4. Проверка выполнимости зависимости в правой части правила на реализации R. Если зависимость невыполнима, то контрпример R найден и конец алгоритма. Иначе переход на шаг 1.

Комментарий. При дополнении недостающих кортежей из R_X в отношение R будет обеспечена выполнимость первой зависимости правила. Такое расширение R увеличит вероятность обнаружения контрпримера на начальных итерациях алгоритма, и не будет содержать дублированных кортежей, если их не было в исходном отношении R. Дополнение недостающих кортежей необходимо делать в конец отношения R, поскольку эти кортежи отсекаются при генерации следующей реализации R.

Далее рассмотрим алгоритм генерации реализаций отношений R. Ранее отмечалось, что в отношении не должно быть дублированных кортежей. Для описания работы алгоритма будем использовать следующие входные переменные: num_t – текущий номер формируемого кортежа ($num\_t = 0$ при входе соответствует началу моделирования, при выходе – окончанию), n_col – количество столбцов (атрибутов) в реализации отношения R, n_tup – количество строк в R (в алгоритме не может быть меньше двух, рекомендовано $k + 1$), $t[n\_tup, n\_col]$ – массив, содержащий реализацию R, min_a – минимальное значение атрибута (равно 0 с учетом значения $Null$, равно 1 без учета значения $Null$), max_a – максимальное значение атрибута (рекомендовано значение 2).

Алгоритм генерации отношений:

```
SUB Gener(num_t, n_col, n_tup, t, min_a, max_a)
IF num_t = 0 THEN
    DO i = 1 TO n_col
        t(1, j) = min_a
        t(2, j) = min_a
    END DO
    num_t = 2
END IF
DO WHILE num_t > 0
    IF Next(num_t, n_col, t, min_a, max_a) THEN
        IF num_t = n_tup THEN
            EXIT SUB
        ELSE
```

```
                num_t = num_t + 1
                DO i = 1 TO n_col
                    t(num_t, j) = t(num_t − 1, j)
                END DO
            END IF
        ELSE
            num_t = num_t − 1
        END IF
END DO
END SUB
```

Алгоритм генерации кортежа:

```
FUNC Next(num_t, n_col, t, min_a, max_a)
i = n_col
DO WHILE i > 0
    IF t(num_t, i) < max_a THEN
        t(num_t, i) = t(num_t, i) + 1
        Next=TRUE
        EXIT FUNC
    ELSE
        IF i > 1 THEN
            DO j = i TO n_col
                t(num_t, j) = min_a
            END DO
            i = i − 1
        ELSE
            Next=FALSE
            EXIT FUNC
        END IF
    END IF
END DO
END FUNC
```

Рассмотренные алгоритмы соответствуют полному перебору всех возможных вариантов реализации отношения R. Ранее была получена оценка количества реализаций без согласования кортежей: d^{nm}, где n – количество столбцов, m – количество кортежей ($k + 1 = m$), d – количество различных значений атрибутов. Для предложенного алгоритма оценка количества реализаций равна количеству сочетаний без повторений из d^n элементов по m. Для сравнения при $n = 7$, $m = 4$ и $d = 2$ получаем $d^{nm} = 268\,435\,456$, тогда как количество сочетаний равно 10 668 000, что более чем в 20 раз меньше.

Поставив в соответствие каждой реализации ее порядковый номер можно вместо полного перебора использовать какие-либо эвристические алгоритмы поиска контрпримера. Однако при решении поставленной задачи это нецелесообразно. Поскольку значительное количество экспериментов с предложенным алгоритмом показало, что если контрпримеры есть, то один из них обнаруживается на начальных итерациях. Если контрпримера нет на начальных итерациях, то ни разу значительное увеличение количества итераций не приводило к искомому результату.

4. Анализ результатов тестирования правил вывода

Для тестирования правил вывода зависимостей соединения с использованием рассмотренного алгоритма рассмотрим следующую зависимость в качестве левой части правила:

$$\bowtie (ABE, BC, ACE, EF, EG, FG).$$

Зависимость имеет шесть компонент, для которых структура связей представлена на рисунке 2. Символами R_i обозначены отношения базы данных, полученные декомпозицией исходного отно-

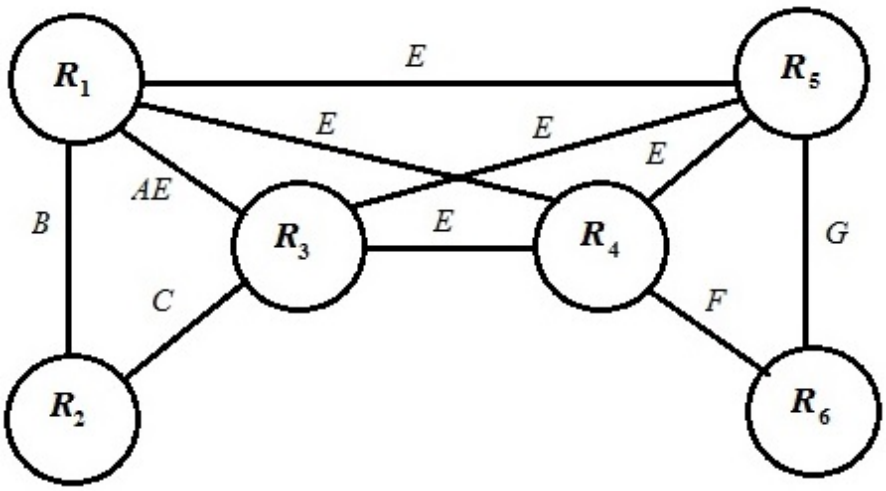

Fig. 2. Relationship Structure of Join Dependency

Рис. 2. Структура связей зависимости соединения

шения $R(ABCDEFG)$ в соответствии с зависимостью соединения: $R_1(ABE)$, $R_2(BC)$, $R_3(ACE)$, $R_4(DEF)$, $R_5(EG)$, $R_6(FG)$. Ребрами на рисунке 2 обозначены общие атрибуты в отношениях, а узлами являются отношения R_i. Заметим, что в соответствующем гиперграфе ситуация будет обратная: ребрами будут отношения R_i, а узлами – атрибуты A, B, C, D, E, F, G.

Далее будем исследовать правила следующего вида:

$$\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (Y_1, Y_2), \tag{4}$$

где Y_1 и Y_2 различные компоненты зависимости в левой части правила. Покажем, что правило (4) не является следствием правил P0 – P4. Правило P0 неприменимо для вывода (4), поскольку левая часть правила P0 содержит пустое множество. Правило P1 неприменимо, поскольку не выполнено условие p11. Правило P2 неприменимо, поскольку в левой части правила (4) при любых указанных Y_1 и Y_2 присутствуют компоненты, для которых не выполнено условие p21. Правило P3 неприменимо, поскольку не совпадает количество компонентов в левой и правой части правила (4). Правило P4 неприменимо, поскольку отсутствие второй зависимости в левой части P4 делает его тривиальным (левая и правая части правила будут совпадать при пустых компонентах второй зависимости).

При тестировании правила $\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, BC)$ по предложенным рекомендациям контрпример был найден на десятой итерации (таблица 6).

Тестирование правила $\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, BC)$ при меньшем количестве кортежей контрпример был найден на девятнадцатой итерации (таблица 7).

При большем количестве итераций контрпример 5 был получен быстрее, чем контрпример 4. Это дает основание для предварительного поиска контрпримера на меньшем количестве кортежей,

Table 6. Counterexample 4 **Таблица 6.** Контрпример 4

$R =$

A	B	C	D	E	F	G
1	1	1	1	1	1	1
1	1	1	1	1	1	2
1	1	1	1	1	2	1
1	1	1	1	1	2	2
1	1	1	1	2	1	1
1	1	1	1	2	1	2
1	1	2	1	1	1	1
1	1	2	1	1	1	2
1	1	2	1	1	2	1
1	1	2	1	1	2	2

Table 7. Counterexample 5 **Таблица 7.** Контрпример 5

$R =$

A	B	C	D	E	F	G
1	1	1	1	1	1	1
1	1	2	1	2	1	1

чем было рекомендовано ранее. Однако в этом случае отсутствие контрпримера не гарантирует его отсутствие на большем количестве кортежей. Подтверждением этому служит тест следующего правила: $\bowtie (AB, BCDEFG, GH) \models \bowtie (ABCE, BEFG, CFGH)$. При начальном количестве кортежей, равном двум, контрпример не был найден (итоговое количество итераций 32 640). При начальном количестве кортежей, равном трем, на итерации 2 273 был найден контрпример (таблица 8).

Table 8. Counterexample 6 **Таблица 8.** Контрпример 6

$R =$

A	B	C	D	E	F	G	H
1	1	1	1	1	1	1	1
1	1	1	1	2	1	2	1
1	1	2	1	1	1	2	1

Вернемся к рассмотрению правил вида (4). Тестирование правил с двумя компонентами в правой части показало, выполнимы (отсутствует контрпример) только следующие правила:

$$\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, DEF),$$

$$\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, EG),$$

$$\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ACE, DEF),$$

$$\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ACE, EG).$$

Для всех остальных правил с двумя компонентами в правой части был найден контрпример. Анализ структуры и состава правил не выявил каких-либо признаков, которые позволили бы отличить невыполнимые правила от выполнимых. Попытка анализа правил с тремя компонентами в правой части также не выявила закономерностей. Ожидалось, что объединение левых частей двух выполнимых правил даст выполнимое правило. Однако для правила $\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, DEF, EG)$ на третьей итерации был найден контрпример (таблица 9). Тогда как при объединении правых частей невыполнимых правил было получено правило $\bowtie (ABE, BC, ACE, EF, EG, FG) \models \bowtie (ABE, BC, ACE)$, для которого контрпример не был найден. Рассмотренные примеры подтверждают

Table 9. Counterexample 7 **Таблица 9.** Контрпример 7

$R =$

A	B	C	D	E	F	G
1	1	1	1	1	1	1
1	1	1	1	1	2	2

необходимость дальнейшего исследования зависимостей соединения с целью корректного построения пятой нормальной формы БД.

5. Использование тестирования при декомпозиции суперключа

Рассмотрим пример проектирования фрагмента БД. Информация в примере относится к предприятию, которое занимается сдачей в аренду изделий различного типа. У каждого типа изделия есть несколько собственников, которые предоставили свои изделия в аренду. Рассмотрим минимальный набор атрибутов: A_1 – инвентарный номер изделия, A_2 – дата выдачи изделия, A_3 – идентификатор клиента, A_4 – плановая дата возврата изделия, A_5 – номер собственника изделия, A_6 – название собственника изделия, A_7 – номер типа изделия, A_8 – название изделия, A_9 – характеристика изделия. Представленное множество атрибутов удовлетворяет множеству функциональных зависимостей: $F = \{A_1A_2A_3 \longrightarrow A_4, A_5 \longrightarrow A_6, A_1 \longrightarrow A_7, A_7 \longrightarrow A_8A_9\}$. В соответствии с алгоритмом синтеза схемы БД [1] будем иметь следующие схемы отношений R_1 – R_4, представленные в таблицах 10 – 13. В отношениях жирным шрифтом выделены первичные ключи. Суперключ для

Table 10. Rental of products (R_1) **Таблица 10.** Выдача изделий в прокат (R_1)

инвентарный номер изделия	**дата выдачи изделия**	**идентификатор клиента**	плановая дата возврата изделия

Table 11. Product owners (R_2) **Таблица 11.** Владельцы изделий (R_2)

номер собственника изделия	название собственника изделия

Table 12. Product Inventory (R_3) **Таблица 12.** Инвентаризация изделий (R_3)

инвентарный номер изделия	номер типа изделия

Table 13. Product description (R_4) **Таблица 13.** Описание изделий (R_4)

номер типа изделия	название изделия	характеристика изделия

множества реализованных зависимостей F состоит из атрибутов: $A_1A_2A_3A_5$. Отношение, сформированное по атрибутам суперключа, представлено в таблице 14. Это отношение содержит аномалию

Table 14. Superkey **Таблица 14.** Суперключ

инвентарный номер изделия	**дата выдачи изделия**	**идентификатор клиента**	**номер собственника изделия**

дополнения-модификации: номера собственников изделия надо повторять столько раз, сколько в БД зарегистрировано изделий данного типа и сколько раз каждое изделие выдавалось в прокат.

На атрибутах суперключа имеет место многозначная зависимость с двумя компонентами в базисе: $A_1 \twoheadrightarrow A_2A_3|A_5$. В соответствии с теоремой Фейджина (Fagin) выполним декомпозицию отношения-суперключа на два отношения: таблицы 15 и 16.

Table 15. Relation 1 **Таблица 15.** Отношение 1

инвентарный номер изделия	дата выдачи изделия	идентификатор клиента

Table 16. Relation 2 **Таблица 16.** Отношение 2

инвентарный номер изделия	номер собственника изделия

Отношение в таблице 15 поглощается отношением R_1 (редукция схемы БД). Во втором отношении осталась аномалия дополнения-модификации: номера собственников изделия надо повторять столько раз, сколько в БД зарегистрировано изделий данного типа. Хотя исчезла аномалия, связанная с выдачей изделия в прокат. Интуитивно ясно, что собственников надо связать с типами изделий. Однако для этого должно существовать формальное обоснование.

Многозначная зависимость $A_1 \twoheadrightarrow A_2A_3|A_5$ эквивалентна зависимости соединения $\bowtie (A_1A_2A_3,\ A_1A_5)$. Рассмотрим правило (не выводимо из P0–P4):

$$\bowtie (A_1A_2A_3,\ A_1A_7,\ A_5A_7) \vDash \bowtie (A_1A_2A_3,\ A_1A_5). \tag{5}$$

Тестирование этого правила с использованием рассмотренных алгоритмов и четырех начальных кортежей ($k + 1 = m$) не выявило контрпримера. При этом было сделано 35 960 итераций (полный перебор вариантов). Получается, что зависимость $\bowtie (A_1A_2A_3,\ A_1A_5)$ выводима, и декомпозицию, полученную по многозначной зависимости, можно заменить декомпозицией, соответствующей левой части правила (5). Итоговая схема БД дополнится только одним отношением R_5 (таблица 17).

Table 17. Own (R_5) **Таблица 17.** Собственность (R_5)

номер типа изделия	номер собственника изделия

Отношения R_1 – R_5 находятся в пятой нормальной форме (5НФ) и обладают свойством соединения без потери информации благодаря реализации зависимостей. Кроме того, ни одно из отношений не содержит ни одной аномалии, и структура БД может расширяться без разрушения существующих отношений. Для составления корректных SQL-запросов к БД необходимо помнить, что в результате декомпозиции суперключ содержится в отношениях R_1, R_3 и R_5.

Заключение

Необходимость разработки технологии тестирования правил вывода стала очевидной при исследовании системы P0 – P4 в работе [8]. Анализ правил вывода, представленный в данной работе, показал, что система P0 – P4 не может считаться полной и требует дальнейшего совершенствования. В данной статье показано, что существуют выполнимые правила, которые не выводимы из известных правил. Несложно заметить, что предложенная технология может быть использована для анализа других видов зависимостей в данных с незначительной коррекцией программного обеспечения.

References

[1] J. Ullman, *Principles of Database Systems.* Stanford University: Computer Science Press, 1980.

[2] D. Maier, *The Theory of Relational Databases.* Rockville: Computer Science Press, 1983.

[3] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases: The Logical Level (1st. ed.)* Addison-Wesley Longman Publishing Co., 1995.

[4] M. Casanova, R. Fagin, and C. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies", *Journal of Computer and System Sciences*, vol. 28, no. 1, pp. 29–59, 1984. DOI: 10.1016/0022-0000(84)90075-8.

[5] H. Köhler and S. Link, "Inclusion dependencies reloaded", in *The 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*, 2015, pp. 1361–1370. DOI: 10.1145/2806416.2806539.

[6] V. S. Zykin and S. V. Zykin, "Analysis of Typed Inclusion Dependences with Null Values", *Automatic Control and Computer Sciences*, vol. 52, no. 7, pp. 638–646, 2018. DOI: 10.3103/S0146411618070258.

[7] E. Sciore, "A Complete Axiomatization of Full Join Dependencies", *Journal of the ACM*, vol. 29, no. 2, pp. 373–393, 1982. DOI: 10.1145/322307.322313.

[8] S. V. Zykin, "Generalization of Derivation Rules for Join Dependencies in Database", *Automatic Control and Computer Sciences*, vol. 55, no. 7, pp. 731–737, 2021. DOI: 10.3103/S0146411621070191.

[9] D. Maier, A. O. Mendelzon, and Y. Sagiv, "Testing implications of data dependencies", *ACM Transactions on Database Systems*, vol. 4, no. 4, pp. 455–469, 1979. DOI: 10.1145/320107.320115.

[10] D. Maier, Y. Sagiv, and M. Yannakakis, "On the Complexity of Testing Implications of Functional and Join Dependencies", *Journal of the ACM*, vol. 28, no. 4, pp. 680–695, 1981. DOI: 10.1145/322276.322280.

[11] P. C. Fischer and D. M. Tsou, "Whether a set of multivalued dependencies implies a join dependency is NP-hard", *SIAM Journal on Computing*, vol. 12, no. 2, pp. 259–266, 1983. DOI: 10.1137/0212015.

[12] C. Beeri and M. Vardi, "Formal Systems for Tuple and Equality Generating Dependencies", *SIAM Journal on Computing*, vol. 13, no. 1, pp. 76–98, 1984. DOI: 10.1137/0213006.

[13] C. Beeri and M. Vardi, "A Proof Procedure for Data Dependencies", *Journal of the ACM*, vol. 31, no. 4, pp. 718–741, 1984. DOI: 10.1145/1634.1636.

[14] M. Gyssens, "On the complexity of join dependencies", *ACM Transactions on Database Systems*, vol. 11, no. 1, pp. 81–108, 1986. DOI: 10.1145/5236.5237.

[15] S. Bell and P. Brockhausen, "Discovery of Data Dependencies in Relational Databases", University of Dortmund, 1995. DOI: 10.17877/DE290R-8395.

[16] J. Kivinen and H. Mannila, "Approximate inference of functional dependencies from relations", *Theoretical Computer Science*, vol. 149, no. 1, pp. 129–149, 1995. DOI: 10.1016/0304-3975(95)00028-U.

[17] M. W. Vincent and M. Levene, "Restructuring Partitioned Normal Form Relations without Information Loss", *SIAM Journal on Computing*, vol. 29, no. 5, pp. 1550–1567, 2000. DOI: 10.1137/S0097539797326319.

[18] S. V. Zykin, "Formation of Hypercube Representation of Relational Database", *Programming and Computer Software*, vol. 32, no. 6, pp. 348–354, 2006. DOI: 10.1134/S0361768806060077.

[19] S. Hartmann, S. Link, and T. Trinh, "Constraint Acquisition You Can Chase but You Cannot Find", in *Conferences in Research and Practice in Information Technology Series*, vol. 79, 2008, pp. 59–68.

[20] F. Marchi, S. Lopes, and J. M. Petit, "Unary and n-ary inclusion dependency discovery in relational databases", *Journal of Intelligent Information Systems*, vol. 32, pp. 53–73, 2009. DOI: 10.1007/s10844-007-0048-x.

[21] X. Hu, M. Qiao, and Y. Tao, "I/O-efficient join dependency testing, Loomis Whitney join, and triangle enumeration", *Journal of Computer and System Sciences*, vol. 82, no. 8, pp. 1300–1315, 2016.

[22] F. M. Malvestuto, "A complete axiomatization of full acyclic join dependencies", *Information Processing Letters*, vol. 68, no. 3, pp. 133–139, 1998. DOI: 10.1016/S0020-0190(98)00148-3.