# tanulmányok

209/1988



MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

Magyar Tudományos Akadémia Számitástechnikai és Automatizálási Kutató Intézete
Исследовательский Институт Вычислительной Техники и Автоматизации
Венгерской Академии Наук

# СИНТАКСИЧЕСКИЕ И СЕМАНТИЧЕСКИЕ СТРУКТУРЫ РЕЛЯЦИОННЫХ ЯЗЫКОВ ЗАПРОСОВ

*ГАЛЯ МЛАДЕНОВА АНГЕЛОВА*

Tanulmányok 209/1988
Труды 209/1988

A kiadásért felelős:

*KEVICZKY LÁSZLÓ*

Диссертация на соискание
ученной степени
кандидата математической наук

Научный руководитель: чл.-корр. АН ВНР

дмн Янош ДЕМЕТРОВИЧ

ISBN 963 311 251 5
ISSN 0324-2951

# СИНТАКСИЧЕСКИЕ И СЕМАНТИЧЕСКИЕ СТРУКТУРЫ РЕЛЯЦИОННЫХ ЯЗЫКОВ ЗАПРОСОВ

## СОДЕРЖАНИЕ

# В В Е Д Е Н И Е

В последние годы очень широкое распространение получили системы управления базами данных (СУБД). Сегодня они уже воспринимаются в качестве базисного программного обеспечения для сохранения и обработки больших объемов информации. Основными элементами СУБД выступают так называемые языки запросов - языки, с помощью которых потребитель может описывать свои информационные потребности к СУБД. Естественно, основная информационная потребность связана с обеспечением доступа к информации в базе данных и с извлечением этой информации из базы данных. (Именно эти требования извлечения информации называются запросами). Большую важность приобретает проблема проверки эквивалентности запросов и проблема оптимизации запросов. Решение этих проблем позволит извлечь информацию из базы данных самым эффективным способом.

В последнее время начинается широкое распространение СУБД реляционного типа. Это связано с более удобными для конечного пользователя интерфейсами - так называемые реляционные языки запросов.

В настоящей работе рассматривается проблема проверки эквивалентности и оптимизации запросов в реляционной базе данных. Так как эти запросы всегда записываются на каком-то реляционном языке, необходимо исследовать отдельные языки запросов и их возможности выражения.

В соответствии с традиционной классификацией, сделанной в [Улл80], реляционные языки можно разбить на две основные

группы:

а) языки, основанные на реляционной алгебре;

б) языки, основанные на вычислении предикатов.

E.F. Codd [Cod71] предложил реляционную алгебру в качестве языка описания обработки, которую нужно совершить над данными с целью извлечения определенного количества информации. Операнды в реляционной алгебре - это отношения с фиксированным числом атрибутов, над которыми можно применять пять основных операций: объединение отношений $R \cup S$, разность отношений $R-S$, декартово произведение $R \times S$, проекция $\pi_{A1,A2,\ldots,Am}(R)$ и селекция $\sigma_F(R)$, где F формула определенного вида. В терминах вышеуказанных пяти основных операций можно выразить например операции пересечение $R \cap S$, $\theta$-соединение $R \bowtie_{[\theta]} S$, естественное соединение $R \bowtie S$ и так далее. Описывая свои требования к базе данных, пользователь должен специфицировать последовательность отдельных операций для обработки данных. Поэтому реляционная алгебра представляет собой **процедурный язык запросов**. Отметим, что такие языки запросов ориентированы прежде всего к пользователям - программистам, потому что пользователь должен представить себе алгоритм обработки данных для того, чтобы извлечь определенное количество информации.

Реляционные языки запросов, основанные на вычислении предикатов, можно рассматривать в качестве языков более высокого уровня, чем реляционную алгебру. В них пользователь задает множество кортежей, которые нужно извлечь из базы данных. Это множество описывается с помощью предиката, которому должны удовлетворять нужные кортежи. Другими словами, описываются не способы обработки данных, а свойства данных. Естественно, при использовании **декларативного языка** этого типа

пользователь не должен вообще знать путем каких операций реляционной алгебры будет получен ответ и в каком порядке их нужно применять. Зато пользователь должен задавать путь доступа между разными реляционными схемами, другими словами, осуществлять так называемую "логическую навигацию".

В практике существуют разные языки запросов, которые обычно включают средства для описания всех требований пользователя базы данных - требования актуализации данных, требования печати данных, а также и некоторые возможности арифметической обработки. Таким образом языки запросов превращаются в языки манипулирования данными. В настоящей работе языками запросов будем называть языки манипулирования данными, в которых включаются только средства для описания способа извлечения информации из базы данных. Обычно эти языки проектированы:

- на основе реляционной алгебры - например сама реляционная алгебра [Cod71], язык ISBL [Улл80];

- на основе вычисления предикатов - например языки ALPHA [Cod71] и QUEL [SWKH76];

- могут быть отнесены к обоим вышеупомянутым типам - например языки SQUARE [BCKH75], SEQUEL и SQL [AC75].

В этой работе мы будем рассматривать проблему проверки эквивалентности и проблему оптимизации запросов для языков, спроектированы на основе вычисления предикатов.

Первые исследования [Cod72a] проблемы эквивалентности запросов в реляционной базе данных начались одновременно с введением реляционной модели. В этих исследованиях вопрос о существовании ограничений (constraints) на данные, участвующие в конкретных отношениях, не рассматривался.

E.F. Codd [Cod72a] показал, что реляционная алгебра и вычисление предикатов имеют одну и ту же объяснительную силу. Это означает, что если с помощью выражений реляционной алгебры

можно описать некоторое требование пользователя, то же самое требование может быть описанным с помощью формул вычислений предикатов, которые интерпретируются как описания требований пользователя.

Известно, что проблема проверки эквивалентности формул в вычислении предикатов алгоритмически неразрешима - не существует алгоритм для проверки эквивалентности формул вычисления предикатов. Отсюда естественным образом вытекает и неразрешимость в общем случае проблемы проверки эквивалентности реляционных выражений. Imielinski и Lipski [IL183] обобщили этот результат. Они показали, что проблема проверки эквивалентности также алгоритмически неразрешима даже и для тех выражений реляционной алгебры, которые содержат только операции: проекцию $\pi$, селекцию $\sigma_F$ (где в F встречается только равенство), соединение $\bowtie$ и разность - .

Основные затруднения в процессе проверки эквивалентности реляционных выражений вытекают из оператора разности [SaYa80]. Вводя понятие таблицы (tableau), Aho, Sagiv и Ullman показали [ASU79], что проверка эквивалентности реляционных выражений является разрешимой проблемой в случае, когда реляционные выражения содержат только проекцию $\pi$, селекцию $\sigma_F$ и соединение $\bowtie$. В [ASU79] показано, что проверка эквивалентности этих выражений является NP-полной [ГДж82]. Рассмотрен и вопрос о проверке эквивалентности запросов при наличии ограничений - в случае [ASU79,ASU79a] при наличии функциональных зависимостей.

В [ASU79] Aho, Sagiv и Ullman выделили класс таблиц - так называемые простие таблицы, при которых алгоритм оптимизации и проверки эквивалентности заканчивает работу в полиномиальном периоде времени.

В [Sag81] и в [MaUl84] показаны еще три класса таблиц, для

которых существует алгоритм проверки эквивалентности и оптимизации в полиномиальном периоде времени.

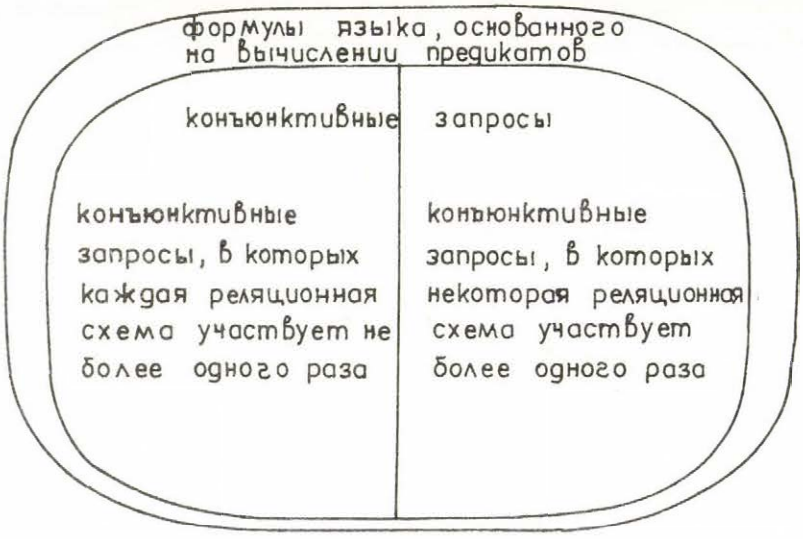
Здесь очень важно отметить, что все вышеописанные классы таблиц, для которых существуют алгоритмы проверки эквивалентности и оптимизации в полиномиальном периоде времени, сформулированы с помощью синтаксиса конкретной таблицы - а именно, где и как встречаются в таблице различные ее переменные и константы. Таким образом, для данного SPJ-выражения мы не можем знать заранее какова будет его таблица. И если для некоторой таблицы не существует алгоритм проверки эквивалентности и оптимизации в полиномиальном периоде времени, мы не можем знать заранее не существует ли эквивалентная ей таблица из вышеописанных классов таблиц, для которых проверка эквивалентности и оптимизация делается в полиномиальном периоде времени.

Таким образом, до сих пор основные теоретические результаты о проверке эквивалентности и оптимизации запросов относятся к классам запросов, сформулированы с помощью терминов реляционной алгебры (или с помощью синтаксиса конкретной таблицы). На фиг. 1 показаны классы этих запросов. Для простоты рассмотрен только класс простых таблиц (в качестве класса запросов, для которых существует алгоритм проверки эквивалентности и оптимизации в полиномиальном периоде времени).

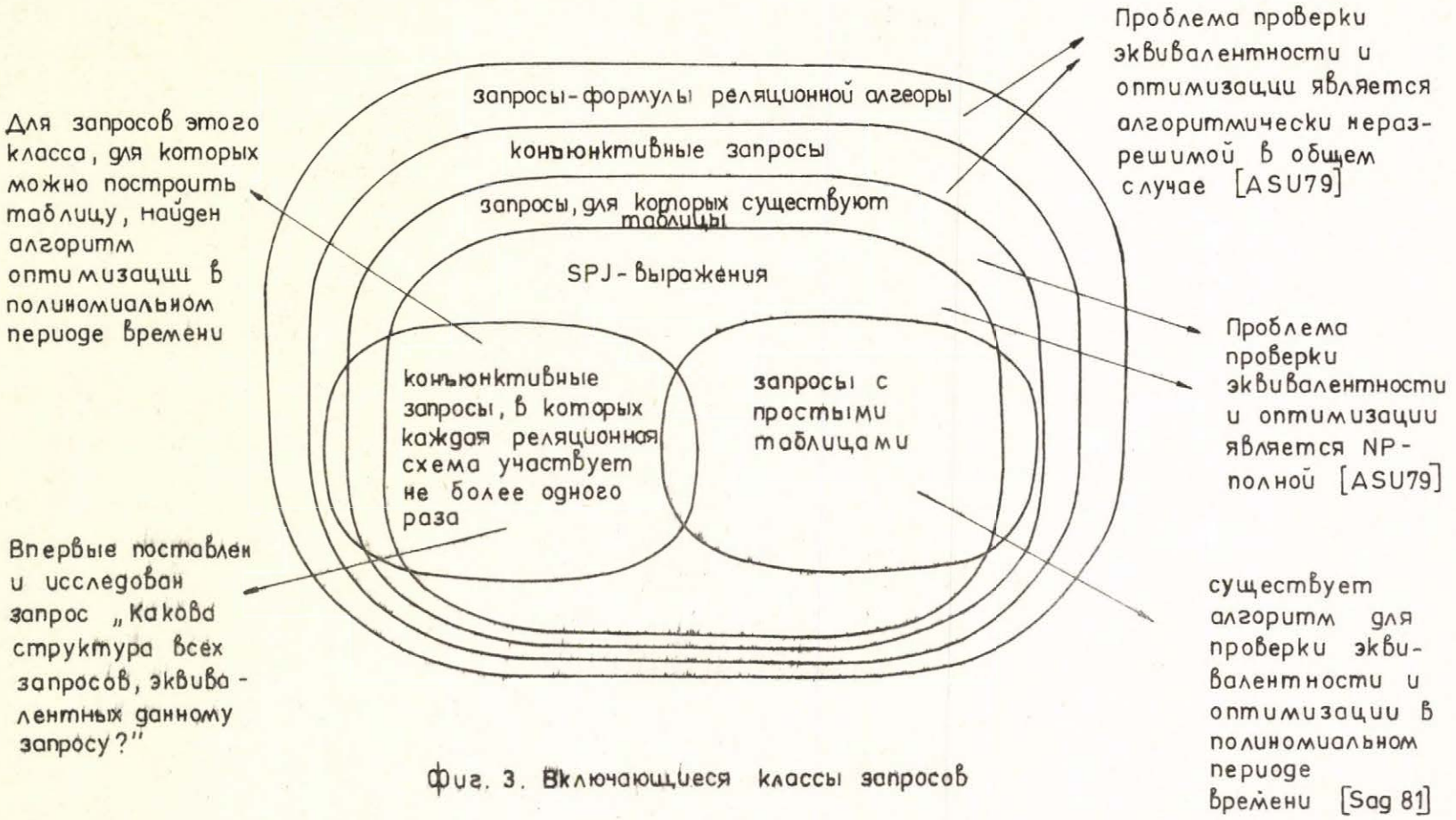
В последнее время декларативные языки запросов (основанные на вычислении предикатов) получают более широкое распространение, чем процедурные языки (основанные на реляционной алгебре) [Ull82]. Следовательно, перед нами встает в первую очередь задача оптимизации запросов, сформулированных на декларативном языке.

Конечно, для языков, основанных на вычислении предикатов, тоже можно рассматривать включающиеся классы запросов - как это


фиг. 1. Классы запросов, сформулированных в терминах реляционной алгебры




фиг. 2. Классы запросов, сформулированных в терминах языка, основанного на вычислении предикатов




Фиг. 3. Включающиеся классы запросов

сделано на фиг. 2. Нужно подчеркнуть, что показанные на фиг. 2 классы запросов не совпадают с классами запросов, показанных на фиг. 1. Таким образом, задавая в некоторой базе данных конъюнктивный запрос, мы не знаем заранее существует ли алгоритм оптимизации в полиномиальном периоде времени для выражения реляционной алгебры, которое реализирует ответ на запрос. В настоящей работе исследованы проблемы проверки эквивалентности и оптимизации запросов для разных классов запросов на декларативном языке.

Приведенная на фиг.3 схема обобщает состояние теоретических исследований в области эквивалентности и оптимизации запросов. Как видно, на схеме отражены разные классы запросов, сформулированных как на декларативных, так и на процедурных языках. (В параграфе 3.4 дан подробный комментарий этой схемы; в параграфах 1.5.4 и 3.4 показаны примеры того, что эти классы запросов содержатся друг в друге.)

Настоящая диссертация ставит следующие цели:

1. Исследовать возможности применения таблиц для оптимизации классов запросов на декларативном языке;

2. Найти условия эквивалентности запросов пользователя к реляционной базе данных, которые позволяли бы эффективную оптимизацию запроса;

3. Исследовать проблему логической навигации в качестве проблемы построения эквивалентных путей доступа, связывающих отдельные реляционные схемы в запросе;

4. Исследовать способы представления путей доступа с помощью синтаксических средств реляционных языков запросов;

5. Рассмотреть возможные преобразования путей доступа в эквивалентные им пути доступа и отсюда возможные преобразования запросов пользователя в эквивалентные им запросы;

6. Исследовать проблему построения интерфейсов высокого

уровня к реляционной базе данных и создать проект интерфейса высокого уровня, в котором могут применяться полученные результаты оптимизации путей доступа в запросах пользователя к данной реляционной базе данных.

Настоящая работа дает ответ на следующие вопросы:

1. Возможно ли применять таблицы для оптимизации запросов на декларативном языке?

2. Существуют ли классы запросов на декларативном языке, которых можно оптимизировать в полиномиальном периоде времени? Какие эти классы запросов и какими методами нужно оптимизировать запросы?

3. Какие синтаксические элементы ("части") запроса "меняются" в процессе оптимизации запроса?

4. Возможно ли найти некоторую классификацию всех запросов, учитывая способ связи синтаксических элементов запроса?

5. Известны разные схемы базы данных - циклические, ациклические. Всегда ли возможно задавать произвольные запросы в разных видах схем? Другими словами, как зависит структура запроса от структуры схемы, к которой этот запрос относится?

6. Существуют ли такие схемы реляционных баз данных, в которых для каждого запроса имеется единственный путь доступа? Если да, то как можно автоматизировать логическую навигацию с целью построения интерфейсов высокого уровня?

7. Возможно ли исследовать проблему эквивалентности запросов, не применяя таблиц? Возможно ли расспознавать эквивалентные запросы с помощью синтаксических структур этих запросов?

В работе получены следующие основные результаты:

1. Рассмотрены конъюнктивные запросы, в которых каждая реляционная схема участвует "не более одного раза" (см.

параграф 1.1). Предложено графическое представление этих запросов. Предложен алгоритм для построения таблицы конъюнктивного запроса, используя его графическое представление (до сих пор алгоритм построения таблиц формулирован в [ASU79] только для SPJ-выражений, но не для запросов на декларативном языке).

Графическое представление этого типа вводится в первый раз.

2. Доказано, что для построенных по предложенному алгоритму таблиц конъюнктивных запросов существует алгоритм оптимизации в полиномиальном периоде времени. При этом не использованы методы доказателств из [ASU79], [Sag81] и [MaUl84] - потому что в настоящей работе показано, что эти методи позволяют найти NP-полные алгоритмы оптимизации. Таким образом предложен оригинальный алгоритм оптимизации конъюнктивных запросов, в которых каждая реляционная схема участвует не более одного раза.

а) доказано, что построенные по предложенному алгоритму таблицы представляют собой минимальную таблицу запросов при сильной эквивалентности;

б) при рассмотрении слабой эквивалентности введено представление конъюнктивных запросов в виде гиперграфов. Таким образом в первый раз запрос представляется в тех же терминах, в которых задаются схемы реляционных баз данных. Показано, что запросы можно обрабатывать средствами, созданными для обработки схем реляционных баз данных, точнее алгоритмом редукции Грэема. Доказано, что для запросов с ациклическим гиперграфом можно получить оптимальный слабоэквивалентный запрос с помощью редукции по Грэему. Для запросов с циклическим гиперграфом сформулировано необходимое и достаточное условие для "открывания" цикла, а также показан алгоритм оптимизации в полиномиальном периоде времени. Таким образом, впервые найден

класс запросов на декларативном языке, имеющих оптимизацию в полиномиальном периоде времени. Нужно отметить, что этот класс запросов является рекурсивно перечислимым множеством (как и класс всех существующих запросов).

3. При исследовании процессов оптимизации таблиц в первый раз учитывается структура связей между отдельными элементами таблиц (потому что например в [ASU79], [Sa981] и [MaUl84] нужно найти отображение определенного вида между символами таблиц несмотря на то, как связаны эти символы между собой).

а) в настоящей диссертации показано, что в случае запросов с ациклическими гиперграфами оптимизацию можно осуществить шаг за шагом, при чем на каждом шагу можно удалить несколько строк из данной таблицы. Этот алгоритм оптимизации заканчивает работу в полиномиальном периоде времени.

б) в настоящей диссертации показано, что в случае запросов с циклическими гиперграфами нельзя оптимизировать запросы шаг за шагом только с помощью операций оптимизации ациклических запросов - потому что в этом случае запросы содержат циклические компоненты и на некотором шаге оптимизации нужно удалить сразу несколько строк из данной таблицы.

Выше мы отметили, что известные до сих пор алгоритмы оптимизации учитывают только синтаксис данной таблицы в процессе ее оптимизации; здесь можем сказать, что предложенные в настоящей работе алгоритмы оптимизации учитывают в некоторой степени и семантику данного запроса.

4. Введен в первый раз новый тип ациклических гиперграфов - так называемые D-ациклические гиперграфи.

5. В первый раз исследован вид конъюнктивных запросов, которые можно задавать к разным видам реляционных схем. В соответствии с видом гиперграфа запроса в первый раз рассмотрены A и D-ациклические запросы и циклические запросы.

Доказано, что в введенных нами D-ациклических реляционных схемах можно задавать только D-ациклические запросы. К остальным видам реляционных схем можно задавать запросы произвольного вида. Таким образом в первый раз выявлен класс реляционных схем, к которым можно задавать только запросы того же самого вида. Можно отметить, что класс D-ациклических схем является рекурсивно перечислимым множеством.

6. Исследован вопрос о виде эквивалентных конъюнктивных запросов. Введено понятие разветвления пути доступа данного запроса. Доказано следующее необходимое и достаточное условие: два запроса в D-ациклической схеме являются эквивалентными тогда и только тогда, когда их пути доступа получаются из "минимального" пути доступа с помощью добавления разветвлений определенного типа.

Условие эквивалентности сформулировано в терминах синтаксиса языка QUEL. Таким образом в первый раз поставлен вопрос: "Как выглядят все запросы, эквивалентные данному запросу?" Настоящая работа дает ответ на этот вопрос в случае конъюнктивных запросов, в которых каждая реляционная схема участвует не более одного раза. Необходимым условием является то, что запросы заданы в некоторой D-ациклической схеме. Таким образом исследованы все пути доступа, эквивалентные данному пути доступа.

7. Исследована проблема о сложности проверки эквивалентности запросов в D-ациклических схемах. В настоящей диссертации показано, что существует алгоритм проверки эквивалентности двух данных запросов, который заканчивает работу в полиномиальном периоде времени. Этот алгоритм построен на основе вышеописанного необходимого и достаточного условия для эквивалентности запросов и проверяет совпадения путей доступа данных запросов.

8. Исследован вопрос об эквивалентности запросов в циклических схемах. Показано достаточное условие для эквивалентности запросов в циклических схемах.

9. Введено понятие "поглощающее ребро гиперграфа" при алгоритме редукции Грэема и показано, что в случае ациклической схемы базы данных поглощающее ребро всегда существует.

10. Используя полученные результаты для проверки эквивалентности и оптимизации таблиц, описан проект интерфейса високого уровня к реляционной базе данных.

Настоящая работа состоит из шести глав. В первой главе даны основные понятия и сделан обзор некоторых последних результатов. Во второй главе рассмотрены понятия ациклической и циклической схемы базы данных. Введено понятие поглощающего ребра ациклического гиперграфа и показано, что такое ребро всегда существует. В третьей главе показан алгоритм построения таблиц запросов для тех конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза. В четвертой главе показано, что для вышеупомянутого класса запросов существует алгоритм оптимизации в полиномиалном периоде времени в случае слабой эквивалентности. В случае сильной эквивалентности показано, что полученные в третьей главе таблицы для этого класса запросов являются минимальными. Введено представление запросов как гиперграфы и исследован вид конъюнктивных запросов в разных схемах реляционных баз данных. Введено понятие D-ациклической схемы реляционной базы данных и показано, что в D-ациклических схемах можно задавать только D-ациклические запросы из вышеупомянутого класса конъюнктивных запросов. В пятой главе рассмотрен вопрос: "Как выглядят все запросы, эквивалентные данному запросу?" Показаны виды всех запросов, эквивалентных данному конъюнктивному запросу в D-

ациклических схемах. Дано достаточное условие для эквивалентности конъюнктивных запросов в циклических схемах. В шестой главе рассмотрены возможности построения интерфейсов высокого уровня к реляционной базе данных.

# Г Л А В А  П Е Р В А Я

# ОСНОВНЫЕ ПОНЯТИЯ И ОБЗОР НЕКОТОРЫХ ПОСЛЕДНИХ РЕЗУЛЬТАТОВ

## 1.1 Основные понятия

Codd [Cod70] предложил реляционную модель баз данных. В классической модели баз данных рассматривается множество атрибутов (attributes) $U$ и множество отношений (relations) над атрибутами $U$. В последние годы введено понятие реляционной схемы. Здесь мы вводим основные понятия, используя [Ull82], [Mai83] и [Дри82].

Пусть дано множество атрибутов

$U=\{A_1,A_2,\ldots,A_n\}$.

Множество $U$ будем называть универсум (universum).

Реляционной схемой (relational scheme) $R$ будем называть конечное множество атрибутов $\{A_{i1},A_{i2},\ldots,A_{im}\}$, где $A_{ij}\in U$ для $1\le j\le m$. Реляционные схемы будем обозначать через $R_1,R_2,\ldots,R_K$.

Каждому атрибуту $A_i$ сопоставляется множество значений - так называемый домен (domain). Домен атрибута $A_i$ будем обозначать через $dom(A_i)$.

Пусть $R$ - реляционная схема, $R=\{A_1,A_2,\ldots,A_n\}$. Отношением $r$ над реляционной схемой $R$ будем называть конечное множество упорядоченных n-ок:

$r = \{\langle a_1 a_2 \ldots a_n \rangle \mid a_1 \in dom(A_1),\ 1 \le i \le n\}$.

Элементы $\langle a_1 a_2 \ldots a_n \rangle$ будем называть кортежами отношения r. Если r - отношение над реляционной схемой $R = \{A_1, A_2, \ldots, A_n\}$, будем обозначать этот факт через $r(R)$ или $r(A_1 A_2 \ldots A_n)$.

Пусть $w = \langle a_1 a_2 \ldots a_n \rangle$ - кортеж отношения $r(A_1 A_2 \ldots A_n)$. Тогда $w[A_i] = a_i$, т.е. через $w[A_i]$ будем обозначать значение атрибута $A_i$ в кортеже w. Если $X = \{A_{i1}, A_{i2}, \ldots, A_{ik}\}$, то

$$w[X] = \langle w[A_{i1}] w[A_{i2}] \ldots w[A_{ik}] \rangle = \langle a_{i1} a_{i2} \ldots a_{ik} \rangle ,$$

где $a_{il} \in dom(A_{il})$ для $1 \le l \le k$.

Отметим, что для каждого конкретного отношения $r(A_1 A_2 \ldots A_n)$ выполнено $\{w[A_i] \mid w \in r\} \subset dom(A_i)$, $1 \le i \le n$.

Отношение является множеством разных кортежей; поэтому можно выявить подмножество атрибутов, значения которых определяют однозначно каждый кортеж. Такое подмножество атрибутов называется кандидат-ключом [Cod72], [BDe79], [Dem78], [Dem79].

Пусть U - данное множество атрибутов. Каждое конечное множество реляционных схем $R = \{R_1, R_2, \ldots, R_k\}$ будем называть схемой базы данных (database scheme) тогда и только тогда, когда

$R_i \subset U$ для $1 \le i \le k$ и $R_1 \cup R_2 \cup \ldots \cup R_k = U$.

(Здесь предполагается, что $R_i \neq U$).

Пусть $R = \{R_1, R_2, \ldots, R_k\}$ - схема реляционной базы данных. Каждое множество конкретных отношений $\{r_1, r_2, \ldots, r_k\}$ соответственно над реляционными схемами $\{R_1, R_2, \ldots, R_k\}$ будем называть состоянием базы данных (database state). Будем

обозначать состояние через $d=\{r_1,r_2,\ldots,r_k\}$.

Пусть $U=R$, т.е. будем рассматривать $U$ как единственную реляционную схему в данной схеме базы данных. Тогда каждое конкретное отношение (состояние) I над схемой $U$ будем называть универсальной реляцией (universal relation, universal instance) [MUV84], [AnZ85].

Определим операции селекции (selection), проекции (projection) и естественного соединения (natural join).

Пусть дана реляционная схема $R=\{A_1,A_2,\ldots,A_n\}$, отношение r над реляционной схемой R и пусть $c\in dom(A_1)$. Пусть $\theta$ - одно из бинарных отношений $=,<,\leq,>,\geq$ и $\neq$. Тогда

- Селекция $A_1\theta c$ (обозначаемая через $\sigma_{A_1\theta c}(r)$ ) представляет собой:

$\sigma_{A_1\theta c}(r) = \{w \mid w\in r \text{ и } w[A_1]\theta c\}$.

Таким образом из отношения r берутся только те кортежи, имеющие такое значение b в атрибуте $A_1$, что $b\theta c$. Селекция $\sigma_{A_1\theta c}(r)$ является отношением над множеством атрибутов $\{A_1,A_2,\ldots,A_n\}$ и следовательно представляет собой подмножество отношения r.

- Проекция $\pi_Y(r)$ представляет собой:

$\pi_Y(r) = \{w[Y] \mid w\in r \text{ и } Y\subset\{A_1,A_2,\ldots,A_n\}\}$, т.е. из всех возможных кортежей отношения r берутся только значения атрибутов множества Y и одинаковые кортежи отождествляются. $\pi_Y(r)$ является отношением над множеством атрибутов Y.

- Естественное соединение $r_1 \bowtie r_2$. Пусть $R_1$ и $R_2$ являются реляционными схемами, $R_1 \cap R_2 \neq \varnothing$ и $r_1(R_1)$, $r_2(R_2)$. Тогда

$r_1 \bowtie r_2 = \{w \mid w$ является кортежем над атрибутами $R_1 \cup R_2$ и существуют кортежи $w_1 \in r_1$ и $w_2 \in r_2$, такие, что $w_1 = w[R_1]$ и $w_2 = w[R_2]\}$.

Таким образом, $r_1 \bowtie r_2$ является отношением над реляционной схемой $R_1 \cup R_2$.

Как было сказано в введении, в настоящей работе рассматриваются проблемы проверки эквивалентности и оптимизации запросов в реляционных базах данных. Введем определение запроса в данной реляционной базе данных.

Пусть $\mathbf{R} = \{R_1, R_2, \ldots, R_k\}$ - схема реляционной базы данных и пусть $d = \{r_1, r_2, \ldots, r_k\}$ - некоторое ее состояние. Пусть для $1 \leq i \leq k$, реляционная схема $R_i$ содержит $n_i$ атрибутов из $U$ и отношение $r_i$ содержит $m_i$ кортежей. Тогда состояние $d$ содержит в разных атрибутах всех реляционных схем

$$m_1 \cdot n_1 + m_2 \cdot n_2 + \ldots + m_k \cdot n_k$$

элементов. Всевозможные подмножества элементов из $d$ будем называть запросами. Видно, что для каждого состояния данной реляционной схеме существуют не больше чем $2^{m1 \cdot n1 + \ldots + mk \cdot nk}$ запросов. Можно показать, что множество всех запросов в разных базах данных является рекурсивно перечислимым множеством.

Рассмотрим способы описания запросов с помощью разных реляционных языков.

E.F. Codd [Cod71] предложил реляционную алгебру в качестве языка описания обработки, которую нужно совершить над данными с

целью извлечения определенного количества информации. Операнды в реляционной алгебре - это отношения с фиксированным числом атрибутов, над которыми можно применять пять основных операций: объединение отношений $r(R) \cup r(S)$, разность отношений $r(R) - r(S)$, декартово произведение $r(R) \times r(S)$, проекция $\pi_{A1,A2,\ldots,Am} r(R)$ и селекция $\sigma_F r(R)$, где F формула определенного вида. В терминах вышеуказанных пяти основных операций можно выразить например операции пересечение $r(R) \cap r(S)$, $\theta$-соединение $r(R) \bowtie_{[\theta]} r(S)$, естественное соединение $r(R) \bowtie r(S)$ и так далее. Описывая свои требования к базе данных, пользователь должен задать отдельные операции для обработки данных и порядок их выполнения. Поэтому реляционная алгебра представляет собой процедурный язык запросов.

В настоящей работе мы будем рассматривать определенный класс формул реляционной алгебры - а именно, класс SPJ-выражений (SPJ-expressions).

ОПРЕДЕЛЕНИЕ 1.1. [ASU79] Формула реляционной алгебры называется SPJ-выражением, если:

а) операнды формулы являются реляционными схемами;

б) операции формулы представляют собой селекцию $\sigma$, проекцию $\pi$ и естественное соединение $\bowtie$. □

Будем обозначать естественное соединение $\bowtie$ и через "J" - как это делается в [ASU79]; с этим обозначением связано и название SPJ- выражения.

Как было сказано в введении, мы будем изучать и представления запросов на декларативных реляционных языках, которые проектированы на основе вычисления предикатов. Основная структура этих представлений - так называемые конъюнктивные

запросы.

ОПРЕДЕЛЕНИЕ 1.2. [Ull82] Конъюнктивным запросом в декларативном реляционном языке называется выражение вида:

$$\{ a_1 a_2 \ldots a_n \mid (\exists b_1)(\exists b_2)\ldots(\exists b_m) : (P_1 \& P_2 \& \ldots \& P_k) \},$$

где $P_l$, $1 \le l \le k$, выражения вида а) или б): (1.1)

а) $R(c_1 c_2 \ldots c_s)$ - это означает, что кортеж $\langle c_1 c_2 \ldots c_s \rangle$ принадлежит отношению над реляционной схемой R. Здесь $c_j$, $1 \le j \le s$, являются константами соответствующего домена или $c_j \in \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$;

б) $c \,\theta\, d$, где c и d - либо константы соответствующих доменов, либо элементы множества $\{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$. Здесь $\theta$ одно из бинарных отношений $<, \le, >$ и $\ge$.

В выражении (1.1) символы $a_1, a_2, \ldots, a_n$ - свободные переменные, а символы $b_1, b_2, \ldots, b_m$ - связанные переменные. □

В настоящей работе для описания конъюнктивных запросов выбран синтаксис одного из реально существующих реляционных языков. Это декларативный реляционный язык QUEL [SWKH76], который проектирован на основе вычисления предикатов.

ОПРЕДЕЛЕНИЕ 1.3. [Ull82] Запросом на языке QUEL называется выражение вида:

```
RANGE OF a1 IS R1
RANGE OF a2 IS R2                                   (1.2)
...
RANGE OF an IS Rn
RETRIEVE (A) WHERE (F)
```

Здесь A называется объектным списком (target list) запроса; F называется квалификацией (qualification) запроса. □

Как сказано выше, в этой работе рассматриваются разные классы конъюнктивных запросов.

На фиг. 2 и фиг. 3 показан класс конъюнктивных запросов, в которых каждая реляционная схема участвует "не более одного раза". Здесь название "реляционная схема, встречающаяся один раз" - условно.

Данная реляционная схема может участвовать в некотором конъюнктивном запросе много раз, но иногда все ее появления могут быть рассмотрены как единственное появление. Например в запросе

$\{ a_1 \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4) : R(a_1, b_1, b_2) \ \& \ R(b_3, c_1, b_4) \}$

реляционная схема R участвует два раза; но этот запрос может быть представлен и следующим образом:

$\{ a_1 \mid (\exists b_1) : R(a_1, c_1, b_1) \}$.

Отметим, что все появления данной реляционной схемы R в некотором конъюнктивном запросе могут быть рассмотрены как единственное появление тогда, когда все свободные переменные и константы запроса, для которых не указана явно их эквивалентность (т.е., не указано например a-c, $a_1$-$a_2$), встречаются в R в столбцах для разных атрибутов. Если для некоторого конъюнктивного запроса q все появления реляционной схемы R могут быть рассмотрены как единственное появление, мы будем называть q **конъюнктивным запросом, в котором реляционная схема R участвует не более одного раза.**

Дальше мы будем изучать класс $K1$ конъюнктивных запросов, в которых каждая реляционная схема участвует не более одного раза.

Легко видеть, что с помощью языка QUEL эти конъюнктивные запросы могут быть рассмотрены как запросы на QUEL вида (1.2), которые содержат по одному RANGE-оператору для каждой реляционной схемы. Эти запросы на QUEL можно представить следующим образом:

$$q:\ \text{RETRIEVE}\ (R_{i1}.A_{i1}, R_{i2}.A_{i2}, \ldots, R_{is}.A_{is}) \qquad (1.3)$$

$$\text{WHERE}\ (R_{j1}.A_{j1}\theta c_1)\ \text{AND}\ (R_{j2}.A_{j2}\theta c_2)\ \text{AND}\ldots\text{AND}\ (R_{jp}.A_{jp}\theta c_p)$$

$$\text{AND}\ (R_{k1}.A_{k1}\theta R_{k2}.A_{k1})\ \text{AND}\ (R_{k3}.A_{k3}\theta R_{k4}.A_{k3})\ \text{AND}\ldots$$

$$\text{AND}\ (R_{kt}.A_{kt}\theta R_{k(t+1)}.A_{kt})\ .$$

Условия квалификации этих запросов задают и путь доступа между отдельными реляционными схемами. Условия квалификации можно разбить на две основные группы:

а) условия, в которых участвует некоторая реляционная схема и константа соответствующего домена -

$(R_{j1}.A_{j1}\theta c_1)\ \text{AND}\ (R_{j2}.A_{j2}\theta c_2)\ \text{AND}\ldots\text{AND}\ (R_{jp}.A_{jp}\theta c_p)$ ;

б) условия, в которых участвуют две реляционные схемы -

$(R_{k1}.A_{k1}\theta R_{k2}.A_{k1})\ \text{AND}\ (R_{k3}.A_{k3}\theta R_{k4}.A_{k3})\ \text{AND}\ldots$

$\text{AND}\ (R_{kt}.A_{kt}\theta R_{k(t+1)}.A_{kt})$.

Как увидим дальше, эти две группы условия выполняют разные роли в квалификации запросов. Поэтому при задании конкретного запроса, мы всегда будем упорядочивать условия квалификации, задавая впервые все условия вида а), а потом все условия вида б).

ОПРЕДЕЛЕНИЕ 1.4. Пусть q - запрос на языке QUEL, записанный в виде (1.3). Запрос q называется <u>связанным</u> тогда и только тогда, когда для каждой пары реляционных схем $R'$ и $R''$, участвующих в (1.3), существуют реляционные схемы $R_1, R_2, \ldots, R_v$

так, что в квалификации запроса q участвуют условия

$(R'.A_1 \theta R_1.A_1)$ AND $(R_1.A_2 \theta R_2.A_2)$ AND...AND $(R_v.A_v \theta R''.A_v)$

для соответствующих атрибутов $A_1, A_2, \ldots, A_v$. □

Легко видеть, что определение 1.4 эквивалентно определению связанности запросов в [WoY76] и [АнД81].

## 1.2 Гиперграфы, соответствующие схемам реляционных баз данных

Одновременно с введением реляционной модели базы данных началось и исследование разных способов организации данных в отдельных массивах (файлах) [Cod70]. Создавались и соответствующие алгоритмы для определения возможной организации данных при наличии заданных требований. Требования задаются с помощью зависимостей данных. Чаще всего применяются функциональные зависимости (см. [Cod72], [DeG81], [DeG81a], [Dem81], [ДПД84]). Существуют алгоритмы декомпозиции в 3НФ с сохранением функциональных зависимостей [Улл80], с помощью которых предлагаются способы разложения в "группы" атрибутов $U$, обеспечивая при этом известные преимущества [Cod72] сохранения данных в отдельных отношениях. Эти алгоритмы в качестве формальной процедуры могут быть использованы для автоматизации проектирования реляционных баз данных. Легко видеть, что для этих алгоритмов всегда существует "выход", который является единственным с точностью до переименования атрибутов и полученных реляционных схем.

В последнее время рассматривается и другой способ разделения атрибутов $U$ на группы, а именно, путем определения $U$ с помощью предикатов [FMU82]. Рассмотрим следующий пример:

ПРИМЕР 1.1. [FMU82] Пусть атрибуты U - C(курс), T(преподаватель), R(аудитория), H(время), S(студент) и G(оценка). При определении универсальной реляции над этими атрибутами пишем:

{ ctrhsg | t преподает c, курс c собирается в r во время h, s получает g по c } .

Будем пользоваться тремя неформально заданными отношениями, которые являются предикатами [FMU82] и которые по нашему мнению имеют смысл в реальном мире: "преподает", "собирается в во время", "получает оценку по".

В этом примере универсальная реляция является множеством тех и только тех кортежей, которые проходят через тест, имплицированный каждым предикатом. Следовательно, универсальную реляцию для этого примера можно записать в виде:

$$\{ ctrhsg \mid P_1(c,t) \& P_2(c,r,h) \& P_3(c,s,g) \} , \qquad (1.4)$$

где $P_1(c,t)$ - предикат "t преподает c",

$P_2(c,r,h)$ - предикат "c собирается в r во время h" и

$P_3(c,s,g)$ - предикат "s получает оценку g по предмету c".

□

Отметим, что определение универсальных реляций с помощью предикатов отражает субъективное представление проектировщика базы данных о реальном мире и следовательно, вряд ли может быть автоматизировано: ясно, однако, что таким образом также можно обеспечить разделение атрибутов на семантические группы и следовательно некоторую степень нормализации.

Введение гиперграфов соответствует описанию универсальной реляции с помощью предикатов.

ОПРЕДЕЛЕНИЕ 1.5. [FMU82] Пусть **E**-конечное множество вершин и **N**-конечное множество ребер, при чем ребра являются непустыми множествами вершин. Тогда упорядоченную пару H= (**E**,**N**) будем называть гиперграфом. □

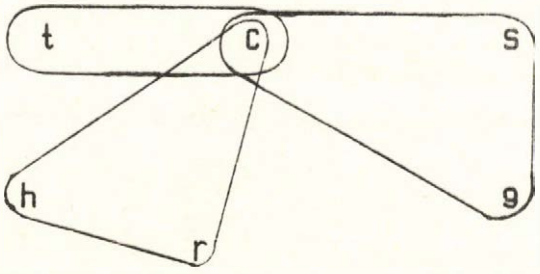
Гиперграф является обобщением обычного графа [Кри78], где каждому ребру соответствуют ровно две вершины.

Каждой универсальной реляции над **U** будем сопоставлять гиперграф следующим образом:

- каждому атрибуту из **U** сопоставляется вершина гиперграфа, соответствующий универсальной реляции;

- каждому предикату в определении универсальной реляции над **U** сопоставляется ребро гиперграфа.

Для универсальной реляции в примере 1.1 определенной с помощью (1.4), получаем гиперграф:




с вершинами соответственно c,t,r,h,s,g и ребрами {c,t}, {c,h,r}, {c,s,g}. □

При задании универсальной реляции с помощью предикатов возможно использовать различные множества предикатов. Тогда видно, что одной универсальной реляции можно сопоставить разные гиперграфы - где каждый гиперграф соответствует отдельному описанию универсальной реляции с помощью предикатов.

Пусть H= (**E**,**N**) - гиперграф и пусть **E**= $\{e_1, e_2, \ldots, e_s\}$. Если

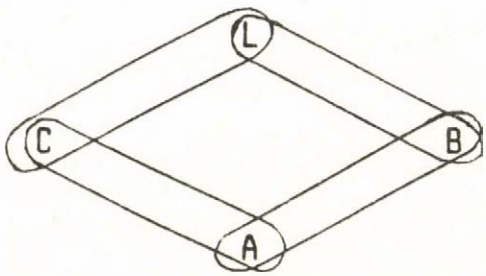
ясно какие вершины участвуют в ребрах $e_1, e_2, \ldots, e_s$, для удобства иногда будем обозначать H через $H = \{e_1, e_2, \ldots, e_s\}$.

## 1.3 Почему универсальная реляция так часто подвергается критике?

Универсальную реляцию можно определить как модель данных, которая в последные годы имеет значительное количество как сторонников, так и противников (о критике ее см. например [Ан87]). Одним из основных недостатков универсальной реляции является то, что для некоторых схем базы данных, классическая модель универсальной реляции предлагает нам довольно слабые средства для описания действительности. Продемонстрируем это на следующем примере.

ПРИМЕР 1.2. Рассмотрим схему базы данных, состоящей из следующих атрибутов: C (пользователь), B (банка), L (заим), A (вклад). Упорядоченные пары CL, CA, LB и AB означают соответственно следующие факты: данный пользователь C берет заим L; данный пользователь C имеет вклад A; каждый заим L берется у некоторого банка B; каждый вклад A помещен в некотором банке B.

Гиперграф, отражающий семантику этой предметной области, выглядит следующим образом:




Требование существования универсальной реляции означает, что каждый пользователь должен иметь и заим, и вклад, причем в

одном и том же банке. Другими словами, в универсальной реляции CLAB для каждого кортежа все значения должны быть заполнены одновременно - что уже является существенным ограничением действительности.

Этот недостаток классической модели универсальной реляции может быть устранен путем допущения нулевых значений в универсальной реляции [Sag81a]. В нашем примере нули (^) могут быть внесены следующим образом:

| C | L | A | B |
|---|---|---|---|
| $c_1$ | $l_1$ | ^ | $b_1$ |
| $c_1$ | ^ | $a_1$ | $b_2$ |

Таким образом мы бы могли описать с помощью универсальной реляции тот факт, что пользователь $c_1$ берет заим $e_1$ от банка $b_1$ и имеет вклад $a_1$ в банке $b_2$ и т.д. □

ОПРЕДЕЛЕНИЕ 1.6. [Sag81a] Модель универсальной реляции с нулями будем называть <u>слабым отношением</u> (weak universal instance). □

В следующей части изложения будем пользоватся слабым отношением везде, где возможно.

## 1.4 Сильная и слабая эквивалентность

После введения универсальной реляции в качестве модели представления данных становится возможным определить понятие сильной и слабой эквивалентности реляционных выражений [ASU79].

ОПРЕДЕЛЕНИЕ 1.7. [ASU79] Пусть даны реляционные схемы $R_1,R_2,\ldots,R_n$. Каждое задание конкретных отношений $r_1,r_2,\ldots,r_n$ соответственно над реляционными схемами $R_1,R_2,\ldots,R_n$ будем называть присваиванием (an assignment). □

ОПРЕДЕЛЕНИЕ 1.8. [ASU79] Пусть $E_1$ и $E_2$ - выражения реляционной алгебры, в которых в качестве операндов участвуют реляционные схемы $R_1,R_2,\ldots,R_n$. Выражения $E_1$ и $E_2$ являются сильно эквивалентными ($E_1 \text{ -- } E_2$), если значения $E_1$ и $E_2$ являются одинаковыми для каждого конкретного присваивания отношений $r_1,r_2,\ldots,r_n$ над реляционными схемами $R_1,R_2,\ldots,R_n$.

□

Для введения понятия слабой эквивалентности будем полагать, что существует универсальная реляция I над $U_{I-1}^{n}R_I$ такая, что конкретные значения отношений $r_1,r_2,\ldots,r_n$ для каждого присваивания являются проекциями этой универсальной реляции I, т.е. $r_I=\pi_{RI}(I)$ для $1\le I\le n$, где I - универсальная реляция над $U_{I-1}^{n}R_I$.

ОПРЕДЕЛЕНИЕ 1.9. [ASU79] Пусть $E_1$ и $E_2$ - выражения реляционной алгебры, в которых в качестве операндов участвуют реляционные схемы $R_1,R_2,\ldots,R_n$. Выражения $E_1$ и $E_2$ являются слабо эквивалентными ($E_1 \text{ --}_w E_2$), если значения выражений $E_1$ и $E_2$ совпадают для каждого конкретного присваивания $r_1,r_2,\ldots,r_n$, где $r_1,r_2,\ldots,r_n$ являются проекциями некоторой универсальной

реляции с атрибутами $\cup_{i=1}^{n} R_i$. □

Очевидно, если $E_1 \text{ -- } E_2$, то $E_1 \text{ --}_w E_2$. Обратное неверно.

ПРИМЕР 1.3. [ASU79] Покажем, что выражения

$$\pi_{AB}(AB \bowtie BC) \quad \text{и} \tag{1.5}$$

$$AB \tag{1.6}$$

являются слабо эквивалентными, не являясь сильно эквивалентными.

Пусть $U = \{A,B,C\}$ - универсум над атрибутами A,B,C и пусть I - конкретная универсальная реляция над $U$. Тогда

$r(AB) = \{ ab \mid (\exists c)$ такое, что $abc \in I\}$;

$r(BC) = \{ bc \mid (\exists a)$ такое, что $abc \in I\}$;

$r(AB) \bowtie r(BC) = \{abc \mid (\exists c')(\exists a')$ такие, что $abc' \in I$ и $a'bc \in I\}$.

Тогда

$$\pi_{AB}(r(AB) \bowtie r(BC)) = \tag{1.7}$$

$$= \{ ab \mid (\exists c)(\exists c')(\exists a') \text{ такие, что } abc' \in I \text{ и } a'bc \in I \} .$$

Выражение (1.6) можно записать следующим образом:

$$\{ ab \mid (\exists c) \text{ такое, что } abc \in I \} . \tag{1.8}$$

Полагая в (1.7) a-a' и c-c' получаем, что выражения (1.7) и (1.8) - одинаковы, следовательно (1.5) и (1.6) - слабо эквивалентны.

Покажем, что выражения (1.5) и (1.6) не являются сильно эквивалентными.

Пусть

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_2$ |

и

| B | C |
|---|---|
| $b_1$ | $c_1$ |
| $b_1$ | $c_2$ |
| $b_3$ | $c_3$ |

Тогда для выражения $AB \bowtie BC$ имеем

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_1$ | $c_2$ |

Следовательно, для выражения $\pi_{AB}(AB \bowtie BC)$ имеем

| A | B |
|---|---|
| $a_1$ | $b_1$ |

Видно, что на этом примере $\pi_{AB}(AB \bowtie BC) \neq AB$.

Следовательно, выражения $\pi_{AB}(AB \bowtie BC)$ и AB - слабо эквивалентны, но не сильно эквивалентны. □

Отметим, что использования нулей в классической модели универсальной реляции (которое усиливает средства выразительности этой модели, как было отмечено в параграфе 1.3) в общем случае не приводит к усилению понятия слабой эквивалентности и превращению его в сильную эквивалентность. При сильной эквивалентности можно допустить и такие присваивания, для которых значения в общих атрибутах отдельных отношений не совпадают (как это имело место в примере 1.3, где значения атрибута B в отношение над AB были $\{b_1, b_2\}$, а значения атрибута B в отношение над BC были $\{b_1, b_3\}$). В отличие от этого случая, все присваивания при слабой эквивалентности должны сохранять одинаковые значения в общих атрибутах отдельных отношений независимо от того, что проекцию можно осуществлять над универсальной реляцией с нулями.

## 1.5 Таблицы запросов и оптимизация в реляционных базах данных

В практике при реализации некоторых реляционных языков делались попытки оптимизировать обработку запросов. В качестве примера можно указать декомпозицию в языке запросов QUEL реляционной системы INGRES [SWKH76,WoY76]. Эта декомпозиция представляет собой стратегию обработки определенного класса запросов. Естественно, при таком практическом подходе не даются доказательства, что полученный запрос является оптимальным по определенному критерию оптимальности и так далее. Другие интересные практические подходы оптимизации описаны например в [CLV85,CeG85].

Теоретические результаты в области оптимизации запросов связаны с понятием таблицы (tableau) [ASU79]. Рассмотрим это понятие более подробно, используя [An986].

### 1.5.1. Определение таблиц запросов

ОПРЕДЕЛЕНИЕ 1.10. [ASU79] Таблицей запроса называется матрица со столбцами, соответствующими атрибутам из $U$ в фиксированном порядке. Первую строку таблицы будем называть резюме таблицы; остальные строки будем называть строками таблицы. Элементы таблицы могут быть символами разного вида:

а) свободные переменные - будем обозначать их через $a, a_1, a_2, \ldots$;

б) связанные переменные - будем обозначать их через $b, b_1, b_2, \ldots$;

в) константы - будем обозначать их через $c, c_1, c_2, \ldots$ . Если константа $c$ появляется в столбце для атрибута $A$, то тогда $c \in dom(A)$;

г) пустые символы (пробелы).

Резюме таблицы содержит только свободные переменные, константы и

пробелы. Каждая переменная может участвовать только в одном столбце таблицы. Если переменная a(b) появляется в столбце атрибута A, то a∈dom(A) (соответственно b∈dom(A) ). Если некоторая свободная переменная появляется в резюме таблицы, она должна появляться по крайней мере еще в одной строке таблицы.

Каждой строке таблицы ставим с правой стороны имя некоторой реляционной схемы $R_J$. Это имя записывается с правой стороны строки как ($R_J$) и называется маркером строки. Здесь $R_J$ - некоторая реляционная схема той базы данных, для которой задается таблица. Если $R_J=\{A_{J1},A_{J2},\ldots,A_{Js}\}$, в каждой строке с маркером ($R_J$) заполнены только столбцы, соответствующие атрибутам $A_{J1},A_{J2},\ldots,A_{Js}$. □

Таблицы введены в [ASU79] как средство представления запросов. Рассмотрим это на следующем примере:

ПРИМЕР 1.4. Рассмотрим следующую базу данных, состоящую из пяти реляционных схем:

ЧАСТЬ (ЧИМЯ,ЧНОМЕР,ЦЕНА)

ПОСТАВЩИК (ПИМЯ,ПНОМЕР,ПАДРЕС,ПГОРОД)

КЛИЕНТ (КИМЯ,КНОМЕР,КАДРЕС,КГОРОД)

ПОСТАВКА (ЧНОМЕР,ПНОМЕР,КНОМЕР,КОЛИЧЕСТВО)

ОБЯЗАННОСТЬ (ЧНОМЕР,ПНОМЕР)

В этой базе сохраняются данные о поставщиках, их клиентах и о поставках разных частей. Отношение ОБЯЗАННОСТЬ дает информацию об обязанностях, присущих каждому поставщику - какой поставщик какие части может поставлять.

В этой базе данных можно рассмотреть следующие примерные запросы:

$q_1$: Найти имена всех поставщиков, живущих в городе $c_1$.

Этот запрос можно представить и следующим образом:

$$\{ \langle a_1 \rangle \mid (\exists b_1)(\exists b_2): \text{ПОСТАВЩИК}(a_1, b_1, b_2, c_1) \}. \qquad (1.9)$$

$q_2$: Найти имена всех поставщиков, поставляющих часть $c_1$ клиентам из города $c_2$.

Этот запрос можно представить с помощью выражения:

$$\{\langle a_1 \rangle \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5)(\exists b_6)(\exists b_7)(\exists b_8)(\exists b_9): \qquad (1.10)$$
$$\text{ПОСТАВЩИК}(a_1, b_1, b_2, b_3) \,\&\, \text{КЛИЕНТ}(b_4, b_5, b_6, c_2) \,\&$$
$$\text{ЧАСТЬ}(c_1, b_7, b_8) \,\&\, \text{ПОСТАВКА}(b_7, b_1, b_5, b_9) \}$$

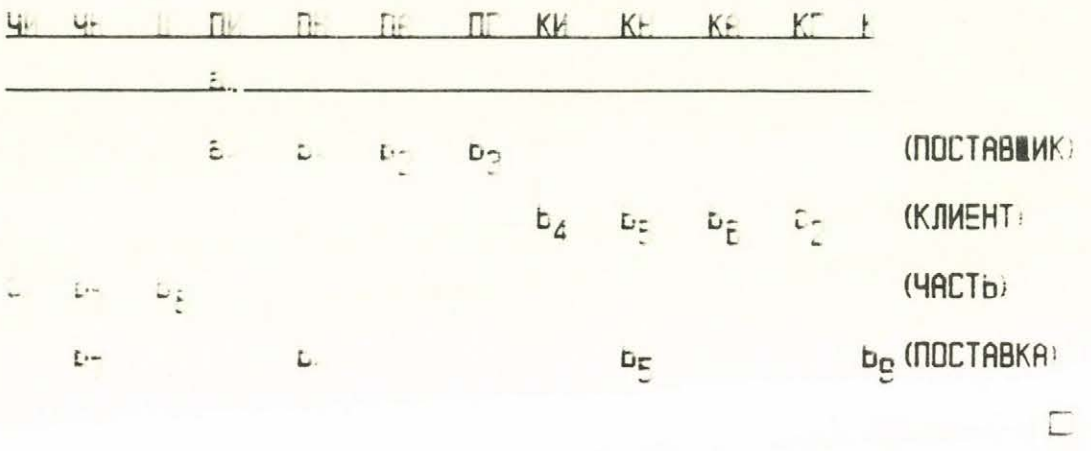
Используя выражение (1.9), запросу $q_1$ сопоставляется таблица $T_1$:

| ПИМЯ | ПНОМЕР | ПАДРЕС | ЦЕНА | |
|---|---|---|---|---|
| $a_1$ | | | | |
| $a_1$ | $b_1$ | $b_2$ | $c_1$ | (ПОСТАВЩИК) |

Для краткости, обозначим атрибуты примерной базы данных следующим образом:

ЧИМЯ - ЧИ, ЧНОМЕР - ЧН, ЦЕНА - Ц, (1.11)
ПИМЯ - ПИ, ПНОМЕР - ПН, ПАДРЕС - ПА, ПГОРОД - ПГ,
КИМЯ - КИ, КНОМЕР - КН, КАДРЕС - КА, КГОРОД - КГ,
КОЛИЧЕСТВО - К.

Тогда используя выражение (1.10), запросу $q_2$ можно сопоставить таблицу $T_{q_2}$:




Пусть $T$ - таблица. Пусть резюме таблицы содержит свободные переменные $a_1, a_2, \ldots, a_s$ и константы $c_1, c_2, \ldots, c_s$ соответственно в столбцах атрибутов $A_1, A_2, \ldots, A_p, A_{p+1}, A_{p+2}, \ldots, A_{p+q}$. Пусть все связанные переменные таблицы $T$ - переменные $\{b_1, b_2, \ldots, b_r\}$. Пусть $T$ содержит $m$ строк с маркерами $R_1, R_2, \ldots, R_m$ (некоторые из маркеров могут быть одинаковыми). Пусть $T$ содержит в строке с маркером $R_1$ символы $\{s_1^1, s_2^1, \ldots, s_{K1}^1\}$, в строке с маркером $R_2$ символы $\{s_1^2, s_2^2, \ldots, s_{K2}^2\}, \ldots,$ в строке с маркером $R_m$ символы $\{s_1^m, s_2^m, \ldots, s_{Km}^m\}$ (здесь символы $s_j^i, 1 \le i \le R_m$, $1 \le j \le \max(K_1, K_2, \ldots, K_m)$, обозначают все свободные переменные, все связанные переменные и все константы таблицы $T$).

ОПРЕДЕЛЕНИЕ 1.11. [ASU79] Пусть $T$ - таблица. Результатом таблицы $T$ будем называть отношение

$$r(T) = \{\langle a_1 a_2 \ldots a_p c_1 c_2 \ldots c_q \rangle \mid a_1 \in dom(A_1), a_2 \in dom(A_2), \ldots, a_p \in dom(A_p),$$
$$c_1 \in dom(A_{p+1}), c_2 \in dom(A_{p+2}), \ldots, c_q \in dom(A_{p+q}) \&$$
$$(\exists b_1)(\exists b_2)\ldots(\exists b_r) : R_1(s_1^1, s_2^1, \ldots, s_{k1}^1) \&$$
$$R_2(s_1^2, s_2^2, \ldots, s_{k2}^2) \& \ldots \& R_m(s_1^m, s_2^m, \ldots, s_{km}^m) \}.$$

□

Можно отметить, что результат таблицы является отношением над теми атрибутами U, которые в резюме данной таблицы содержат свободные переменные и константы.

ПРИМЕР 1.5. Для таблицы $T_1$ из примера 1.4 результатом является отношение

$r_1(T_1)$ - {$\langle a_1 \rangle$ | $a_1$ ∈ ПИМЯ и существует значение $b_1$ атрибута ПНОМЕР и $b_2$ атрибута ПАДРЕС так, что кортеж $\langle a_1 b_1 b_2 c_1 \rangle$ принадлежит отношению ПОСТАВЩИК }.

□

ПРИМЕР 1.6. Для таблицы $T_2$ из примера 1.4 результатом является отношение

$r_2(T_2)$ - {$\langle a_1 \rangle$ | $a_1$ ∈ ПИМЯ и существует $b_1$ ∈ ПНОМЕР, $b_2$ ∈ ПАДРЕС, $b_3$ ∈ ПГОРОД, $b_4$ ∈ КИМЯ, $b_5$ ∈ КНОМЕР, $b_6$ ∈ КАДРЕС, $b_7$ ∈ ЧНОМЕР, $b_8$ ∈ ЦЕНА, $b_9$ ∈ КОЛИЧЕСТВО так, что $\langle a_1 b_1 b_2 b_3 \rangle$ ∈ ПОСТАВЩИК & $\langle b_4 b_5 b_6 c_2 \rangle$ ∈ КЛИЕНТ & $\langle c_1 b_7 b_8 \rangle$ ∈ ЧАСТЬ & $\langle b_7 b_1 b_5 b_9 \rangle$ ∈ ПОСТАВКА }.

□

ПРИМЕР 1.7. Рассмотрим таблицу T:

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | |
|---|---|---|---|---|---|
| | $a_1$ | | $a_2$ | | |
| $b_1$ | $a_1$ | $b_2$ | | | $(R_1)$ |
| | | $b_2$ | $a_2$ | $b_3$ | $(R_2)$ |
| | $b_4$ | | $b_5$ | $c_1$ | $(R_3)$ |

Строки этой таблицы показывают, что $R_1 = \{A_1, A_2, A_3\}$, $R_2 = \{A_3, A_4, A_5\}$ и $R_3 = \{A_2, A_4, A_5\}$.

Результат таблицы T можно записать следующим образом: $r(T) = \{\langle a_1 a_2 \rangle \mid a_1 \in A_2, a_2 \in A_4$ и существует $b_1 \in A_1, b_2 \in A_3, b_4 \in A_2$ и $b_5 \in A_4$ так, что $\langle b_1 a_1 b_2 \rangle \in R_1, \langle b_2 a_2 b_3 \rangle \in R_2$ и $\langle b_4 b_5 c_1 \rangle \in R_3 \}$.

Легко представить запись конъюнктивного запроса, для которого составлена таблица T. Этот запрос выглядит следующим образом:

$$\{a_1 a_2 \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5) : R_1(b_1 a_1 b_3) \wedge R_2(b_2 a_2 b_3) \wedge R_3(b_4 b_5 c_1) \}.$$ □

Дальше мы не будем отмечать над таблицей имена атрибутов из U, если это ненужно.

### 1.5.2. Построение таблиц по данным SPJ-выражениям

Рассмотрим алгоритм построения таблицы для данного SPJ-выражения [ASU79].

Таблица данного SPJ-выражения содержит столбцы для всех атрибутов из U. При построении таблицы заполняются только столбцы соответствующие атрибутам реляционных схем, участвующих в данном SPJ-выражении [ASU79].

Значение каждого SPJ-выражения является отношением и его можно рассматривать в качестве ответа некоторого конъюнктивного запроса. Следовательно для данного SPJ-выражения мы можем построить таблицу, представляющую собой запрос,ответом которого является данное SPJ-выражение. Так как SPJ-выражение является формулой [Cod72a] и ее можно строить индуктивным образом [ASU79], то таблицу SPJ-выражения также можно строить индуктивным образом.

АЛГОРИТМ 1.1 [ASU79] Построение таблицу по данному SPJ-выражению.

ВХОД: SPJ-выражение.

МЕТОД: Строим таблицу для данного SPJ-выражение по числу операции $\pi$,б и J, которые содержатся в данном SPJ-выражении.

Пусть E является SPJ-выражением, которое содержит 0 операция - т.е. оно не содержит ни одной из операция $\pi$,б и J. Тогда E=R, где R - некоторая реляционная схема $R=\{A_{i1},A_{i2},\ldots,A_{is}\}$.Тогда таблица состоит из резюме и еще из одной строки с маркером (R). В столбцах резюме, соответствующих атрибутам реляционной схемы R, находятся свободные переменные. Другие столбцы резюме пустые. В столбцах строки (R), соответствующих атрибутам реляционной схемы R, находятся те же самые свободные переменные, которые находятся в резюме. Другие столбцы строки заполнены разными связанными переменными.

В качестве примера можно рассмотреть построеные таблицы для реляционной схемы ЧАСТЬ на фиг. 1.2.

Допустим, что уже построили таблицу для данного SPJ-выражения, которое содержит n операция $\pi$,б и J. Будем строить

таблицу для SPJ-выражения, которое содержит n+1 операций $\pi$,$\sigma$ и J.

Пусть E является SPJ-выражением, которое содержит n+1 операций $\pi$,$\sigma$ и J. Тогда имеет место одна из следующих трех возможностей:

а) $E=\pi_X(E_1)$, где $X\subset\{A_1,A_2,\ldots,A_k\}$ и $E_1$ является SPJ-выражением, которое содержит не больше чем n операций $\pi$,$\sigma$ и J;

б) $E=\sigma_{A1-c}(E_1)$, где $A_1\in\{A_1,A_2,\ldots,A_k\}$, $c\in dom(A_1)$ и $E_1$ является SPJ-выражением, которое содержит не больше чем n операций $\pi$,$\sigma$ и J;

в) $E=E_1\bowtie E_2$, где $E_1$ и $E_2$ являются SPJ-выражениями, которые содержат не больше чем n операций $\pi$, $\sigma$ и J.

Покажем как строится таблица для каждого из этих случаев:

а) Предположим, что $E=\pi_X(E_1)$ и пусть $T_1$ - таблица для $E_1$. Таблицу T для E строим из таблицы $T_1$ следующим образом: в резюме $T_1$ ставим "пустые символы" в столбцы, непринадлежащие X. Так получаем резюме таблицы T. Во всех остальных строках $T_1$ для столбцов, непринадлежащих X, разные свободные переменные заменяются разными связанными переменными.

В качестве примера можно рассмотреть на фиг. 1.2 построение таблицы для выражения $E=\pi_{ПИМЯ}(E_1)$, где

$E_1$ = ОБЯЗАННОСТЬ $\bowtie$ $\sigma_{ЦЕНА-50}$(ЧАСТЬ) $\bowtie$ ПОСТАВЩИК.

б) Предположим, что $E=\sigma_{A1-c}(E_1)$. Пусть $T_1$ - таблица для $E_1$. Таблицу T для E можно получить из $T_1$ следующим способом:

- если столбец $A_1$ в резюме $T_1$ - пустой, то выражение E не

имеет смысла и тогда таблица $T=\varnothing$;

- если в столбце $A_1$ в резюме $T_1$ имеется константа $c_1$ и $c_1=c$, то $T=T_1$. Если $c_1 \neq c$, считаем, что нельзя сделать селекцию $\sigma_{A1=c}$ и тогда $T=\varnothing$;

- если в столбце $A_1$ в резюме $T_1$ имеется свободная переменная a, то таблица T получается от таблицы $T_1$ путем замещения a через c, независимо от того, в каком месте встречается a в $T_1$.

В качестве примера можно рассмотреть на фиг. 1.2 построение таблицы для выражения $\sigma_{ЦЕНА=50}$ (ЧАСТЬ).

в) Предположим, что $E=E_1 \bowtie E_2$ и $T_1$ и $T_2$ - соответственно таблицы для $E_1$ и $E_2$. Без потери общности можно предположить, что множества связанных переменных $T_1$ и $T_2$ не пересекаются и если в одних и тех же столбцах в строках резюме $T_1$ и $T_2$ фигурируют свободные переменные, то они являются одинаковыми. Таблицу T для E конструируем следующим способом: если в резюме $T_1$ и $T_2$ на одном и том же месте фигурируют разные константы, то $T=\varnothing$. В противном случае строками T являются строки $T_1$ и $T_2$, причем резюме T образовано из резюме $T_1$ и $T_2$ как следует. Если в данном столбце $A_1$ фигурируют:

- константа c в резюме одной из таблиц $T_1$ и $T_2$, то в резюме T ставится c и везде свободная переменная другой таблицы заменяется константой c;

- свободная переменная в одной из таблиц или в обеих, то в резюме T ставится та же переменная;

- пустые символы в обеих таблицах $T_1$ и $T_2$, то в таблицу T

ставим так же пустой символ.

В качестве примера можно рассмотреть построение на фиг.1.2 таблицы для выражения ОБЯЗАННОСТЬ$\bowtie$б$_{\text{ЦЕНА-50}}$(ЧАСТЬ).

ВЫХОД: Таблица T для входного SPJ-выражения. □

ПРИМЕР 1.8. Проиллюстрируем процесс конструирования таблицы для SPJ-выражения на следующем запросе для базы данных из примера 1.4:

$q_3$: Найти имена поставщиков, поставляющие части ценой 50.

Ответ запроса можно представить с помощью SPJ-выражения:

$$\text{п}_{\text{ПИМЯ}}((\text{ОБЯЗАННОСТЬ}\bowtie\text{б}_{\text{ЦЕНА-50}}(\text{ЧАСТЬ}))\bowtie\text{ПОСТАВЩИК}) \qquad (1.12)$$

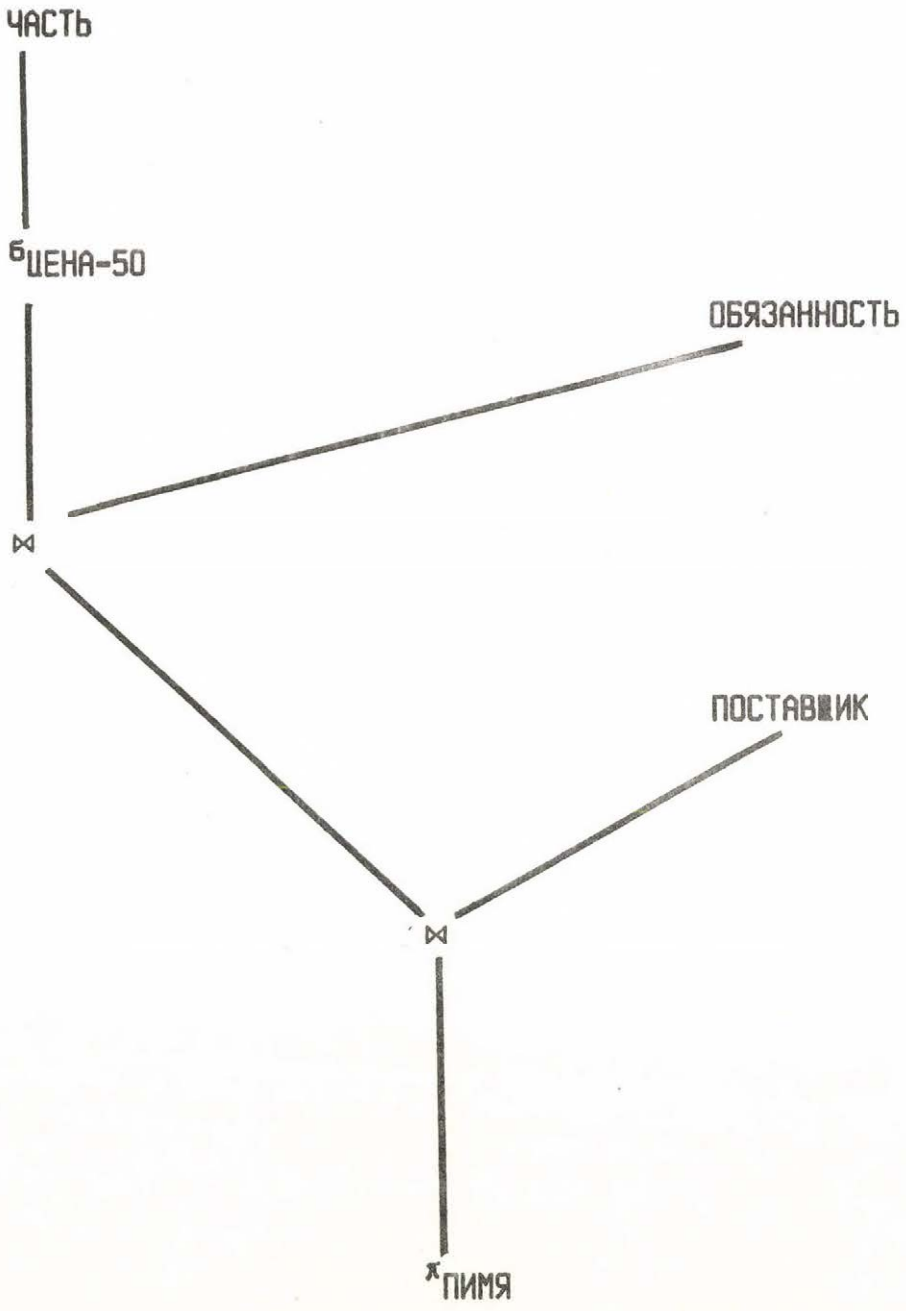
□

Дерево разбора [Улл80] этого выражения дано на фиг. 1.1.

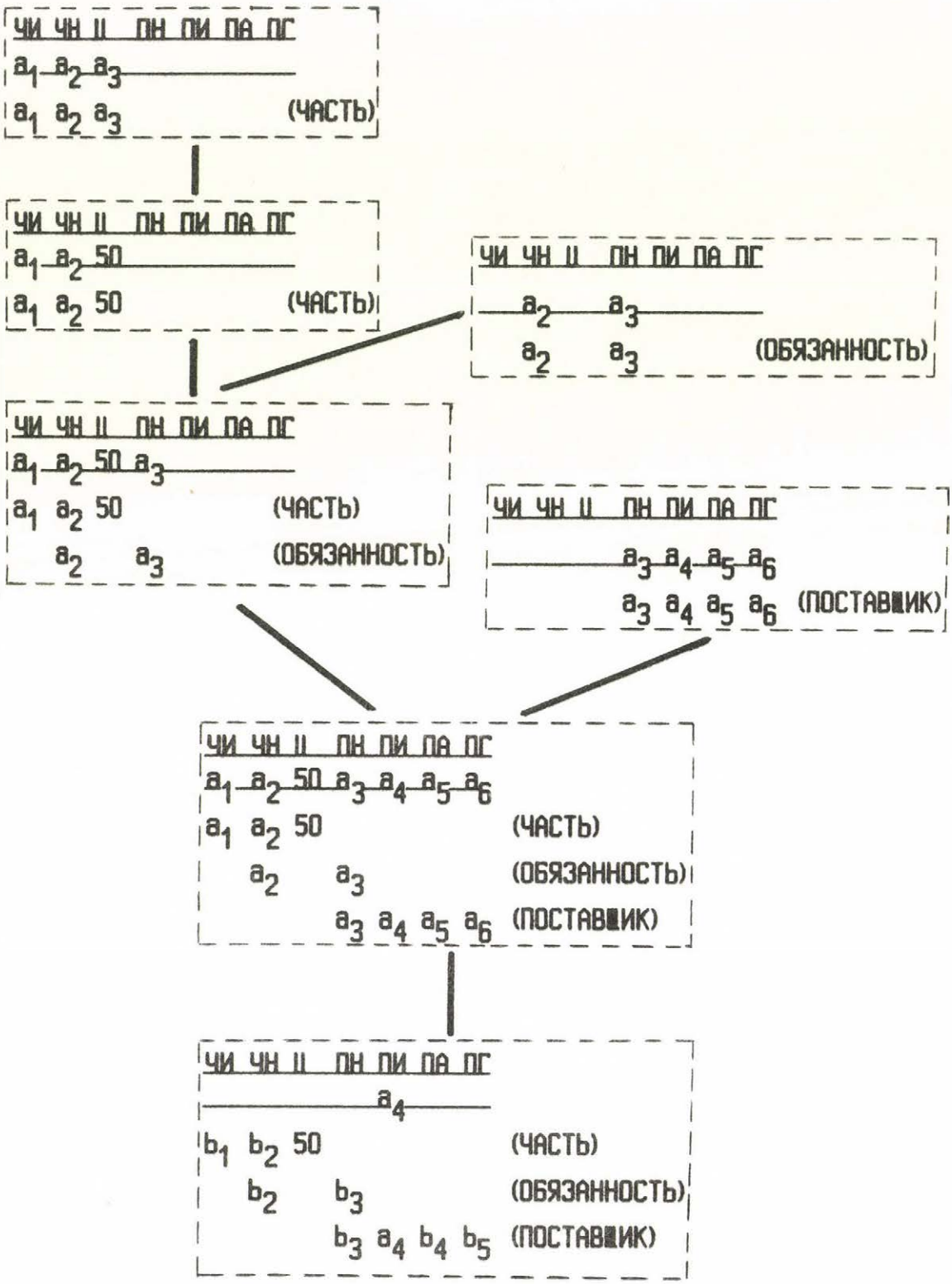
На фиг. 1.2 представляется последовательность конструирования таблицы для данного SPJ-выражения (1.12). Во всех таблицах записаны атрибуты, участвующие в реляционных схемах ОБЯЗАННОСТЬ, ЧАСТЬ и ПОСТАВЩИК.

В таблицах на фиг. 1.2 атрибуты обозначены только через две буквы - как это сделано в (1.11). Все связанные переменные, встречающиеся только один раз, замещены пустыми символами.

Расположение отдельных таблиц на фиг. 1.2 соответствует расположению элементов в дереве разбора на фиг. 1.1. Различные таблицы отделены одна от другой с пунктирной линией.




Фиг.1.1. Дерево разбора для SPJ-выражения (1.12).




Фиг.1.2. Конструирование таблицы для SPJ-выражения (1.12).

### 1.5.3. Эквивалентность и минимизация таблиц

Как указано в [ASU79] и в [ASU79a], понятие таблица вводится с целью исследования эквивалентности запросов, полагая, что таким образом запрос легче поддается формализации. Использование понятия таблицы в качестве средства оптимизации запросов основывается на следующих определениях.

Пусть $d=\{r_1,r_2,\ldots,r_n\}$ представляет собой состояние базы данных, т.е. множество отношений над реляционными схемами $\{R_1,R_2,\ldots,R_n\}$. Тогда результат, сопоставленный данной таблице T с помощью вышеописанной интерпретации (см. определение 1.11), будем обозначать через $T(d)$. Естественно, отношение $T(d)$ является различным для различных состояний d.

ОПРЕДЕЛЕНИЕ 1.12. [ASU79] Будем говорить, что $T_1 \subset T_2$, если для каждого d выполняется $T_1(d) \subset T_2(d)$. □

ОПРЕДЕЛЕНИЕ 1.13. [ASU79] $T_1$ и $T_2$ являются сильно эквивалентными ($T_1 \text{--} T_2$) тогда и только тогда, когда $T_1 \subset T_2$ и $T_2 \subset T_1$. □

В основе определения эквивалентности двух данных таблиц лежит понятие отображения [ASU79] между символами и строками таблиц.

ОПРЕДЕЛЕНИЯ 1.14. [ASU79] Пусть $T_1$ и $T_2$ - таблицы. Отображение h символов $T_1$ в символы $T_2$ называется содержащим отображением (containment mapping), если:

а) h отображает символы резюме $T_1$ в символы резюме $T_2$;

б) h отображает символы любой строки $T_1$ в символы строки $T_2$ с таким же маркером, как у строки из $T_1$. При этом h сохраняет

значения всех констант. □

Если h отображает все символы данной строки в символы другой строки, говорим, что h отображает всю данную строку в другую. Таким образом будем рассматривать h как отображение одного символа в другой и как отображение одной строки в другую.

ТЕОРЕМА 1.1. [ASU79] Пусть $T_1$ и $T_2$ - таблицы. $T_2 \subset T_1$ тогда и только тогда, когда существует содержащее отображение h из $T_1$ в $T_2$. □

Следовательно, $T_1$ и $T_2$ являются сильно эквивалентными ($T_1 \text{--} T_2$) тогда и только тогда, когда существует содержащее отображение $h_1$ из $T_1$ в $T_2$ и содержащее отображение $h_2$ из $T_2$ в $T_1$.

ПРИМЕР 1.9. Рассмотрим таблицы $T_1$ и $T_2$:

$T_1$: $a_1$———— и $T_2$: $a_1$————

| | | | | | |
|---|---|---|---|---|---|
| $a_1$ | $b_1$ | (R) | $a_1$ | $b_1$ | (R) |
| | | | $b_2$ | $b_3$ | (R) |
| | | | $b_4$ | $b_5$ | (R) |

Сразу видно, что таблицы $T_1$ и $T_2$ - сильно эквивалентны, так как существуют содержащие изображения $h_1$ из $T_1$ в $T_2$ и $h_2$ из $T_2$ в $T_1$: $h_1(a_1)-a_1$, $h_1(b_1)-b_1$ и $h_2(a_1)-a_1$, $h_2(b_1)-b_1$, $h_2(b_2)-a_1$, $h_2(b_3)-b_1$, $h_2(b_4)-a_1$, $h_2(b_5)-b_1$. При этом $h_1$ и $h_2$ сохраняют маркеры всех строк. □

ОПРЕДЕЛЕНИЕ 1.15. Пусть T - таблица. Минимальной таблицей

для таблицы T будем называть таблицу, которая содержит минимальное число строк и которая эквивалентна таблице T. □

Процесс нахождения минимальной таблицы для таблицы T будем называть оптимизацией таблицы T.

В [ASU79] показано, что задача нахождения минимальной таблицы для данной таблицы T является NP-полной при сильной эквивалентности.

В рассматриваемых до сих пор таблицах маркеры показывают что данная строка берется из некоторого отношения. Требование рассматривать строки таблиц вместе с их маркерами не в силе, если предположить существование универсальной реляции I над универсумом $U=R_1 \cup R_2 \cup \ldots \cup R_n$, где I такая, что $r_i=\pi_{R_i}(I)$ для $1 \le i \le n$.

Это предположение известно под именем "предположение существования универсума" (universal instance assumption) [MUV84]. В таком случае каждая строка таблиц запросов снабжена маркером U, т.е. каждая строка берется из универсума. Таким образом не нужно учитывать от куда взялась каждая строка и маркеры могут быть пропущены. Тогда при оптимизации таблицы возможно исчезновение всех строк с маркерами ($R_S$) для некоторого отношения $r_S$, которые присутствовали в первоначальном представлении таблицы. Предположение о существовании универсума ведет к определению понятия слабой эквивалентности между таблицами запросов.

ОПРЕДЕЛЕНИЕ 1.16. [ASU79] Пусть $d=\{r_1,r_2,\ldots,r_n\}$ - состояние базы данных такое, что $r_i=\pi_{R_i}(I)$, $1 \le i \le n$, где I - некоторая универсальная реляция над $U=R_1 \cup R_2 \cup \ldots \cup R_n$.

Таблицы $T_1$ и $T_2$ являются слабо эквивалентными ($T_1 \sim_w T_2$) тогда и только тогда, когда $T_1(d) \subset T_2(d)$ и $T_2(d) \subset T_1(d)$ для каждого состояния d, в котором все отношения $r_i, 1 \le i \le n$ являются проекциями некоторой универсальной реляции I над U. □

ТЕОРЕМА 1.2. [ASU79] Пусть $T_1$ и $T_2$ - таблицы. $T_1 \sim_w T_2$ тогда и только тогда, когда существуют содержащее отображение $h_1$ от символов $T_1$ в символы $T_2$ и содержащее отображение $h_2$ от символов $T_2$ в символы $T_1$ такие, что $h_1$ и $h_2$ не учитывают маркеры всех строк. □

ПРИМЕР 1.10. Предположим, что для базы данных из примера 1.4 выполнено предположение о существовании универсума. Ищем ответ на следующий запрос:

$q_4$: Найти имена всех поставщиков, поставляющие (или уже поставившие) части, количество которых 500.

Следующее SPJ-выражение реализует ответ на этот запрос:

$$\pi_{ПИМЯ}(ПОСТАВЩИК \bowtie ОБЯЗАННОСТЬ \bowtie \sigma_{КОЛИЧЕСТВО=500}(ПОСТАВКА)) \qquad (1.13)$$

Этому запросу соответствует таблица $T_3$:

$T_3$:

| ПИ | ПН | ПА | ПГ | ЧН | КН | К | |
|---|---|---|---|---|---|---|---|
| $a_1$ | | | | | | | |
| $a_1$ | $b_1$ | | | | | | (ПОСТАВЩИК) |
| | $b_1$ | | $b_2$ | | | | (ОБЯЗАННОСТЬ) |
| | $b_1$ | | $b_2$ | | | 500 | (ПОСТАВКА) |

При описании таблицы $T_3$ пропущены столбцы атрибутов, в которых участвуют связанные переменные, встречающиеся только один раз в строках таблицы $T_3$.

Рассмотрим таблицу $T_4$:

$T_4$: ПИ ПН ПА ПГ ЧН КН К

$a_1$ ———————————————

$a_1$ $b_1$ (ПОСТАВЩИК)

$b_1$ 500 (ПОСТАВКА)

Сразу видно, что таблицы $T_3$ и $T_4$ не сильно эквивалентны, потому что $T_4$ не содержит строку с маркером (ОБЯЗАННОСТЬ). Но если предположим существование универсальной реляции, тогда не нужно учитывать маркеры в таблицах $T_3$ и $T_4$. Видно, что $T_3$ и $T_4$ – слабо эквивалентны, так как существуют содержащее отображение $h_1$ символов строк $T_3$ в символы строк $T_4$ и содержащее отображение $h_2$ символов строк $T_4$ в символы строк $T_3$. $h_1$ отображает первую строку $T_3$ в первую строку $T_4$, вторую строку $T_3$ во вторую строку $T_4$ и третью строку $T_3$ во вторую строку $T_4$; $h_2$ отображает первую строку $T_4$ в первую строку $T_3$ и и вторую строку $T_4$ в третью строку $T_3$.

Легко видеть, что $T_4$ является минимальной слабо эквивалентной таблицей для таблицы $T_3$.

Таблица $T_4$ представляет выражение:

$$\pi_{\text{ПИМЯ}}(\text{ПОСТАВЩИК} \bowtie_{\text{б}} \sigma_{\text{КОЛИЧЕСТВО-500}}(\text{ПОСТАВКА})). \qquad (1.14)$$

Таким образом ясно, что выражения (1.13) и (1.14) являются слабо эквивалентными. Нужно отметить, что здесь предположение о существовании универсальной реляции является существенным - потому что таблицы $T_3$ и $T_4$ не являются сильно эквивалентными.

□

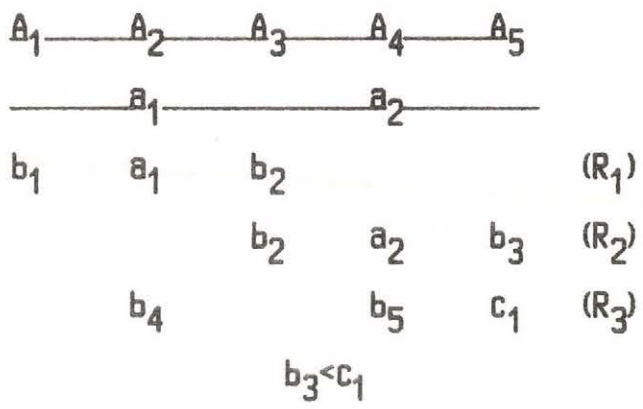
### 1.5.4. Расширение определения таблицы

В [ASU79] рассмотрены только таблицы для тех SPJ-выражений, в которых операция J является естественным соединением. В [Ull82] дано расширенное определение таблицы.

Понятие таблицы в [Ull82] введено как в определение 1.10, но к таблице можно добавить список выражений вида $s_1 \theta s_2$, где $s_1, s_2$ - символы таблицы, а $\theta$ одно из отношений $<$, $\leq$, $=$, $>$ и $\geq$. Эти выражения записываются под строками таблицы. Таким образом формируется список ограничений, который тоже рассматривается как часть таблицы.

В [Ull82] введены и соответствующие расширенные определения для понятия содержащего отображения и эквивалентности двух заданных таблиц.

ПРИМЕР 1.11. Рассмотрим таблицу T из примера 1.7, добавляя к ней ограничение $b_3 < c_1$:




Результат таблицы T можно записать следующим образом:

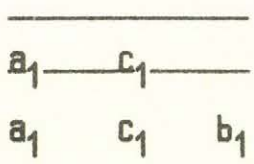
$r(T) = \{\langle a_1 a_2\rangle \mid a_1 \in A_2, a_2 \in A_4$ и существует $b_1 \in A_1, b_2 \in A_3, b_4 \in A_2$ и $b_5 \in A_4$ так, что $\langle b_1 a_1 b_2\rangle \in R_1, \langle b_2 a_2 b_3\rangle \in R_2,$ $\langle b_4 b_5 c_1\rangle \in R_3$ и $b_3 < c_1 \}$.

Легко представить запись конъюнктивного запроса, для которого составлена таблица T. Этот запрос выглядит следующим образом:

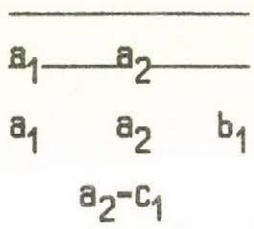
$\{a_1 a_2 \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5) : R_1(b_1 a_1 b_3) \ \& \ R_2(b_2 a_2 b_3)$ $\& \ R_3(b_4 b_5 c_1) \ \& \ (b_3 < c_1) \}$. □

Как видно из определения 1.11, если некоторая таблица содержит константы в резюме, ее результат нельзя интерпретировать как конъюнктивный запрос вида (1.1) - потому что в выражении (1.1) слева от символа "|" встречаются только свободные переменные. Расширенное определение таблицы в [Ull82] позволяет нам сопоставлять конъюнктивный запрос вида (1.1) каждой таблице. Для констант $c_1, c_2, \ldots, c_q$ из определения 1.11 вводим новые свободные переменные $a_{p+1}, a_{p+2}, \ldots, a_{p+q}$, которые записываем, соответственно, слева от "|" в результате таблицы и добавляем к ограничениям таблицы новые ограничения $a_{p+1} = c_1, a_{p+2} = c_2, \ldots, a_{p+q} = c_q$.

ПРИМЕР 1.12. Если используем только определение таблицы из [ASU79], то тогда таблице




нельзя сопоставить конъюнктивный запрос. С помощью определения таблицы из [Ull82] эту таблицу можно записать как




Тогда этой таблице сопоставляется конъюнктивный запрос
$\{a_1a_2 \mid R(a_1,a_2,b_1) \& (a_2=c_1)\}$.

□

Дальше мы будем использовать определение таблицы из [Ull82]. Будем полагать, что для всех запросов, для которых существуют таблицы, существуют также конъюнктивные запросы. Эти конъюнктивные запросы записываются с помощью результата таблиц путем введения новых свободных переменных и соответствующих ограничений.

### 1.5.5. Сложность алгоритма оптимизации таблиц и простые таблицы

Как показано в [ASU79], алгоритм оптимизации таблиц с помощью поиска содержащих отображений является NP-полным.

В [ASU79] показано также, что для одного класса таблиц - так называемых простых таблиц, существует алгоритм оптимизации, который заканчивает работу в полиномиальном периоде времени.

ОПРЕДЕЛЕНИЕ 1.17. [ASU79] Таблица T является простой, если каждый ее столбец обладает свойством "если некоторая связанная переменная встречается в этом столбце больше одного раза, то тогда ни один другой символ не участвует в этом столбце больше, чем один раз". □

Как показано в [ASU79], эквивалентность простых таблиц может устанавливаться за период времени $s^3t^2$ (где s - число строк и t - число столбцов простых таблиц).

В [Sag81] показаны еще два класса таблиц, для которых существует алгоритм проверки эквивалентности в полиномиальном периоде времени:

а) таблицы, содержащие в каждой строке не больше одной связанной переменной, которая участвует и в другой строке таблицы;

б) таблицы, для которых каждая их строка "покрыта" не больше чем одной другой строкой таблицы (строка таблицы $T_1$ "покрывает" [Sag81] строку таблицы $T_2$, когда эти две строки содержат одни и те же свободные переменные и константы и если константы из строки $T_1$ участвуют в резюме $T_1$, то они участвуют и в резюме $T_2$).

Здесь нужно отметить, что все вышеописанные классы таблиц, для которых существуют алгоритмы проверки эквивалентности и оптимизации в полиномиальном периоде времени, сформулированы с помощью синтаксиса конкретной таблицы - а именно, где и как

встречаются в таблице разные ее переменные и константы. Таким образом, для данного SPJ-выражения мы не можем знать заранее какова будет его таблица. И если для некоторой таблицы не существует алгоритма проверки эквивалентности и оптимизации в полиномиальном периоде времени, мы не можем знать заранее не существует ли эквивалентная ей таблица из вышеописанных классов таблиц, для которых проверка эквивалентности и оптимизация делается в полиномиальном периоде времени.

В [MaUl84] Maier и Ullman выявляют класс простых таблиц, которые соответствуют определенному классу гиперграфов. Они показывают, что эти таблицы могут быть оптимизированны с помощью алгоритмов, созданных для обработки гиперграфов (конечно, тоже в полиномиальном периоде времени).

В [ASU79] показано, что существуют таблицы, которые не могут быть построены ни из одного SPJ-выражения. Конечно, для таких таблиц алгоритм оптимизации тоже является NP-полным.

Для всех остальных запросов (для которых не существуют таблицы) проблема проверки эквивалентности и оптимизации является алгоритмически неразрешимой [ASU79].

От вышесказанного становиться ясным, с какими трудностями можем столкнуться при реализации процедуры оптимизации запросов в реляционной базе данных. Можно рассматривать оптимизацию запросов в качестве важной функции базы данных, так как времена ответа двух эквивалентных запросов могут сильно отличаться. Следовательно, мы можем поставить себе целью разработать процедуры, которые по запросу пользователя находили бы оптимальный (самый "дешевый") запрос, эквивалентный данному запросу. К сожалению ясно, что оптимизация сама по себе является дорогостоящим процессом - в случае, когда она вообще возможна. По этой причине большинство существующих прикладных систем исспользуют стандартные модификации и статистика, чтобы

редуцировать "цену" запросов, и не делают попытки найти более дешевые эквивалентные запросы.

# Г Л А В А  В Т О Р А Я

# ЦИКЛИЧЕСКИЕ И АЦИКЛИЧЕСКИЕ СХЕМЫ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

В этой главе рассматриваются понятия ациклической схемы и циклической схемы реляционной базы данных, которыми мы будем пользоваться дальше. Эти понятия введены в [BFMMUY81] с целью дать ответ на вопрос - при каких условиях каждое состояние базы данных, которое является попарно совпадающим (pairwise consistent), является также сплошь совпадающим (Join consistent). Этот вопрос поставлен в 1976-ом году, но его ответ дан в [BFMMUY81] в 1981-ом году с введением понятия циклической и ациклической схемы базы данных.

В параграфе 2.1 даны основные определения попарного и сплошного совпадения. В параграфе 2.2 рассмотрен алгоритм редукции гиперграфов, который в настоящей работе используется для содержательного определения понятия ацикличности (вместо формального определения ациклического гиперграфа, использованного например в [Fag83] и в [Fag83a]). В параграфе 2.3 исследованы некоторые свойства гиперграфов в процессе редукции и доказано существование так называемого поглощающего ребра для ациклических гиперграфов.

## 2.1 Попарное и сплошное совпадения

Дадим основные определения.

ОПРЕДЕЛЕНИЕ 2.1. [Mai83] Состояние базы данных над реляционными схемами $R_1, R_2, \ldots, R_n$ называется попарно совпадающим, если значения одинаковых атрибутов в отношениях над схемами $R_1, R_2, \ldots, R_n$ совпадают. □

ПРИМЕР 2.1. Пусть дана схема реляционной базы данных $\mathbf{R} = \{ABC, BCD, AD\}$ и ее состояние

$r_1(ABC)$ - | A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

, $r_2(BCD)$ - | B | C | D |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

, $r_3(AD)$ - | A | D |
|---|---|
| 0 | 1 |
| 1 | 0 |

.

Покажем, что это состояние - попарно совпадающее.

Каждый атрибут для каждой схемы имеет значения {0,1}; поэтому значения одинаковых атрибутов в отношениях $r_1(ABC)$, $r_2(BCD)$ и $r_3(AD)$ совпадают.

Из определении 2.1 вытекает, что это состояние схемы $\mathbf{R} = \{ABC, BCD, AD\}$ является попарно совпадающим. □

ОПРЕДЕЛЕНИЕ 2.2. [Mai83] Состояние базы данных над реляционными схемами $R_1, R_2, \ldots, R_n$ называется сплошь совпадающим, если все отношения этого состояния являются проекциями некоторой универсальной реляции над атрибутами $\bigcup_{i=1}^{n} R_i$. □

ПРИМЕР 2.2. Рассмотрим состояние из примера 2.1, отыскивая такое состояние универсальной реляции над {ABCD}, что $r_1, r_2$ и

$r_3$ являлись проекциями $r_1=\pi_{ABC}(ABCD)$, $r_2=\pi_{BCD}(ABCD)$, $r_3=\pi_{AD}(ABCD)$.

Покажем, что $r_1(ABC)$, $r_2(BCD)$ и $r_3(AD)$ не являются сплошь совпадающими.

Отношение $r_1(ABC)$ содержит кортеж (0,0,0) и $r_2(BCD)$ содержит кортеж (0,0,0). Отсюда вытекает, что универсальная реляция над {ABCD} должна содержать кортеж (0,0,0,0). Следовательно, если $r_1(ABC)$, $r_2(BCD)$ и $r_3(AD)$ являлись бы сплошь совпадающими, то отношение $r_3(AD)$ должно было содержать кортеж (0,0) - что противоречит содержанию $r_3(AD)$.

И так, показано, что $r_1(ABC)$, $r_2(BCD)$ и $r_3(AD)$ не являются сплошь совпадающими.

Рассмотрим другое состояние для схемы базы данных {ABC,BCD,AD}. Пусть

$r_1(ABC)$ -

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

, $r_2(BCD)$ -

| B | C | D |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

, $r_3'(AD)$ -

| A | D |
|---|---|
| 0 | 0 |
| 1 | 1 |

.

Легко видеть, что $r_1(ABC)$, $r_2(BCD)$ и $r_3'(AD)$ являются сплошь совпадающими. Универсальная реляция над {ABCD}, из которой получаются проекции $r_1(ABC)$, $r_2(BCD)$ и $r_3'(AD)$, имеет следующие кортежи:

$r(ABCD)$ -

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |

и тогда $r_1(ABC)=\pi_{ABC}(r)$, $r_2(BCD)=\pi_{BCD}(r)$, $r_3'(AD)=\pi_{AB}(r)$.

□

Очевидно, сплошное совпадение имплицирует попарное совпаденое; пример 2.2 показывает, что обратное неверно.

## 2.2. Алгоритм редукции гиперграфов

Рассмотрим следующий алгоритм, который называется алгоритмом редукции Грэема (Graham reduction algorithm) [Gra80].

АЛГОРИТМ 2.1. [FVa84] Алгоритм редукции гиперграфа данной схемы реляционной базы данных.

ВХОД: Гиперграф данной схемы реляционной базы данных записан в виде строк следующим образом: каждому ребру гиперграфа отводится одна строка записи; при этом одинаковые атрибуты отдельных реляционных схем располагаются всегда один под другим.

ОПЕРАЦИЯ 1. Зачеркнуть имена всех атрибутов, которые появляются только один раз в входной записи;

ОПЕРАЦИЯ 2. Эсли какая-то строка с атрибутами $\{A_1, A_2, \ldots, A_k\}$ целиком содержит другую строку с атрибутами $\{A_{i1}, A_{i2}, \ldots, A_{ip}\}$, т.е. $\{A_{i1}, A_{i2}, \ldots, A_{ip}\} \subset \{A_1, A_2, \ldots, A_k\}$, зачеркнуть строку с атрибутами $\{A_{i1}, A_{i2}, \ldots, A_{ip}\}$.

МЕТОД: Применять операции 1 и 2 в произвольном порядке, сколько раз возможно.

ВЫХОД: Пустая запись - когда после применения операции 1 и 2 все символы входной записи зачеркнуты;

Непустая запись - когда после применения операции 1 и 2 не могут быть зачеркнуты все символы входной записи. □

Операции 1 и 2 алгоритма редукции не добавляют новые символы к входной записи, а только зачеркивают символы с входной записи. Имея ввиду конечности входной записи видно, что алгоритм 2.1 всегда заканчивает работу. Алгоритм 2.1 работает в полиномиальном периоде времени.

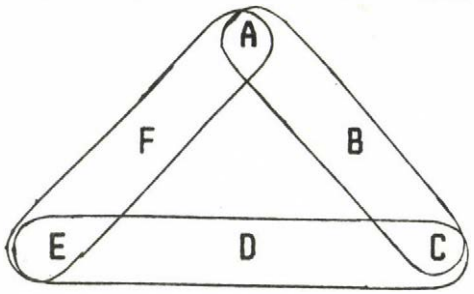
ТЕОРЕМА 2.1. [BFMY83] Гиперграф является ациклическим тогда и только тогда, когда алгоритм редукции заканчивает работу над этим гиперграфом с пустой записью. □

Теорема 2.1 позволяет нам ввести следующее определение:

ОПРЕДЕЛЕНИЕ 2.3. Пусть H - гиперграф, над которым алгоритм 2.1 заканчивает работу с пустой записью. Тогда H будем называть ациклическим гиперграфом. □

Проиллюстрируем алгоритм редукции на следующем примере.

ПРИМЕР 2.3. Пусть дана схема реляционной базы данных {ABC}, {CDE}, {AEF}. Ее гиперграф выглядит следующим образом:




(2.1)

Для применения алгоритма редукции нужно записать этот гиперграф как следует:

A B C

C D E

A E F

Над этой записью применяем операцию 1 из алгоритма редукции и получаем

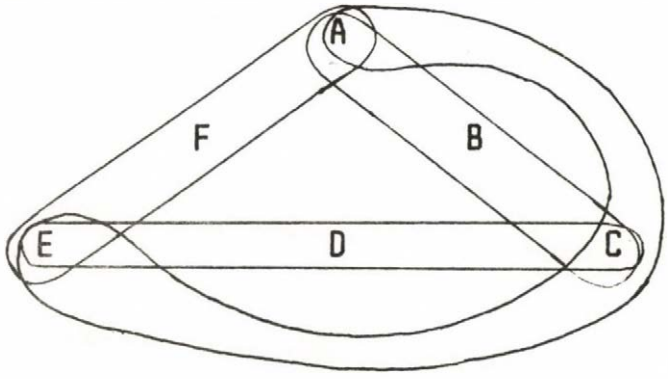
A C

C E

A E

Эта запись является выходом алгоритма редукции, потому что дальнейшее применение как операции 1, так и операции 2 алгоритма 2.1 невозможно. Следовательно, гиперграф (2.1) является циклическим. □

ПРИМЕР 2.4. Пусть дана схема реляционной базы данных {ABC}, {CDE}, {AEF}, {ACE} в соответствии с гиперграфом




(2.2)

Покажем, что гиперграф (2.2) является ациклическим.

Гиперграф (2.2) можно представить в следующем виде:

A B C

C D E

A E F (2.2′)

A C E

Применяем операцию 1 алгоритма редукции над записью (2.2′) и получаем

A C

C E (2.3)

A E

A C E

Применяем три раза операцию 2 из алгоритма 2.1 соответственно для строк $\{A,C\} \subset \{A,C,E\}$, $\{C,E\} \subset \{A,C,E\}$, $\{A,E\} \subset \{A,C,E\}$, и получаем строку

A C E (2.3′)

Применяем операцию 1 из алгоритма 2.1 над записью (2.3′) и получаем пустую запись. Следовательно, гиперграф (2.2) является ациклическим. □

Отметим, что гиперграф (2.2), являясь ациклическим, содержит циклическое подмножество (2.1). Для обычных графов это не верно [Кри78].

Оказывается, что алгоритм редукции дает нам необходимое и достаточное условие для ответа на вопрос "Когда каждое состояние базы данных, которое является попарно совпадающим, является так же и сплошь совпадающим?". Следующие теоремы дают ответ на этот вопрос.

ТЕОРЕМА 2.2. [BFMY83] Пусть схема реляционной базы данных является ациклической. Каждое ее состояние является попарно совпадающим тогда и только тогда, когда это состояние - сплошь совпадающее. □

Следовательно, для ациклических схем баз данных, попарное совпадение эквивалентно сплошном совпадению.

ТЕОРЕМА 2.3. [BFMY83] Если схема реляционной базы данных является циклической, тогда существует ее состояние, которое является попарно совпадающим, но не является сплошь совпадающим. □

Примеры 2.1 и 2.2 показывают такое состояние для схемы {ABC},{BCD},{AD} (которая является циклической).

СЛЕДСТВИЕ 2.1.[BFMY83] Из теоремы 2.2 и 2.3 вытекает, что попарное совпадение эквивалентно сплошном совпадению тогда и только тогда, когда схема базы данных является ациклической. □

Формальное определение понятия ациклического гиперграфа дано в [BFMMUY81, BFMY83, Fag83] и тоже в [MaUl84].

## 2.3 Некоторые замечания об ацикличности

Докажем некоторые свойства ациклических схем, которыми будем пользоваться далее [Анг87].

Будем использовать запись гиперграфов в виде строк, которая представляет собой вход в алгоритме редукции.

ОПРЕДЕЛЕНИЕ 2.4. Изолированной вершиной гиперграфа H называется такая вершина, которая участвует только в одном ребре гиперграфа H. □

Изолированные вершины участвуют также только в одной строке записи гиперграфа H в виде строк и отстраняются от этой записи путем применения операции 1 из алгоритма редукции.

ОПРЕДЕЛЕНИЕ 2.5. Пусть $H=\{e_1,e_2,\ldots,e_p\}$ - гиперграф с ребрами соответственно $e_1,e_2,\ldots,e_p$. Тогда вершины, которые участвуют в хотя бы двух ребрах гиперграфа H, будем называть связанными вершинами. □

Если $H=\{e_1,e_2,\ldots,e_p\}$ - гиперграф, то через $e_1,e_2,\ldots,e_p$ будем обозначать и ребра гиперграфа H, и строки записи для гиперграфа H в виде строк. Будем обозначать множество всех связанных вершин в данной строке $e_i$ через $con(e_i)$. Следовательно, для каждой строки e (для каждого ребра e) гиперграфа H, мы разделяем вершины строки (ребра) в двух непересекающихся множествах: множество con(e) и множество изолированных вершин.

Отметим, что если запись $\{e_1,e_2,\ldots,e_s\}$ содержит строки, состоящие только из изолированных вершин, то тогда после применения операции 1 алгоритма редукции в полученной записи участвуют и пустие строки. Далее мы не будем отмечать существования пустых строк, если это не нужно.

ОПРЕДЕЛЕНИЕ 2.6. Будем говорить, что ребро $e_1=\{A_1,A_2,\ldots,A_n\}$ гиперграфа H <u>поглощает</u> ребро $e_2=\{A'_1,A'_2,\ldots,A'_k\}$ этого гиперграфа, если $con(\{A'_1,A'_2,\ldots,A'_k\})\subset con(\{A_1,A_2,\ldots,A_n\})$. □

Будем обозначать этот факт через "$\succ$", т.е. $e_1\succ e_2$.

Очевидно реляция "$\succ$" является реляцией частичной упорядоченности.

Пусть для двух строк $\{A_1,A_2,\ldots,A_n\}$ и $\{A'_1,A'_2,\ldots,A'_k\}$ выполнено $\{A_1,A_2,\ldots,A_n\}\succ\{A'_1,A'_2,\ldots,A'_k\}$. Тогда в процессе применения алгоритма редукции можно применять операцию 1 над изолированными вершинами в $\{A_1,A_2,\ldots,A_n\}$ и в $\{A'_1,A'_2,\ldots,A'_k\}$, а

потом и операцию 2 над этими строками. При этом строка, содержащая вершин $con\{A'_1,A'_2,\ldots,A'_k\}$, исчезает с входной записи. Таким образом, каждая пара строк $e_1 \succ e_2$ может быть редуцирована до получения строки $con(e_1)$.

ПРИМЕР 2.5. Рассмотрим изолированные и связанные вершины гиперграфа (2.2) в примере 2.4.

Вершины B,D и F являются изолированными для этого гиперграфа. Все остальные вершины этого гиперграфа - т.е. вершины A,C и E, являются связанными. При этом выполнено: $con(\{A,B,C\}) = \{A,C\}$, $con(\{C,D,E\}) = \{C,E\}$, $con(\{A,E,F\}) = \{A,E\}$ и $con(\{A,C,E\}) = \{A,C,E\}$. □

Определения 2.4 и 2.5 введены для записи гиперграфа, которая является входом в алгоритм редукции (потому что ясно, что некоторые связанные вершини гиперграфа могут превратиться в изолированными после применения операции 2 алгоритма редукции гиперграфов). Обобщим определения 2.4 и 2.5 для каждого шага алгоритма 2.1.

ОПРЕДЕЛЕНИЕ 2.7. Пусть дана входная запись гиперграфа H соответственно со строками $\{e_1,e_2,\ldots,e_s\}$. Запись $\{e'_1,e'_2,\ldots,e'_k\}$ будем называть <u>редукцией</u> записи $\{e_1,e_2,\ldots,e_s\}$, если запись $\{e'_1,e'_2,\ldots,e'_k\}$ получена из записи $\{e_1,e_2,\ldots,e_s\}$ путем применения операции 1 возможное число раз и одного применения операции 2 алгоритма редукции гиперграфов. □

Пусть операция 2 алгоритма редукции применяется J раз над

входной записью. Полученную запись будем называт редукцией входной записи на J-том уровне.

ОПРЕДЕЛЕНИЕ 2.8. Пусть дана запись гиперграфа $H=\{e_{l1},e_{l2},\ldots,e_{lm}\}$, которая является редукцией входной записи на J-том уровне. Все вершины в записи $\{e_{l1},e_{l2},\ldots,e_{lm}\}$, которые участвуют только в одном ребре множества $\{e_{l1},e_{l2},\ldots,e_{lm}\}$, будем называть <u>изолированными вершинами (на J-том уровне редукции)</u>; остальные вершины будем называть <u>связанными вершинами (на J-том уровне редукции)</u>. □

ПРИМЕР 2.6. Рассмотрим изолированные и связанные вершины на разных уровнях редукции для гиперграфа (2.3).

Рассмотрим запись (2.3) из примера 2.4:

A C
C E
A E
A C E

На первом уровне редукции этой записи можно получить запись

C E
A E
A C E

В этой записи все вершины связанны на первом уровне редукции.

На втором уровне редукции можно получит запись

A E (2.4)
A C E

В этой записи вершина C является изолированной на втором уровне редукции, а вершины A и E являются связанными на втором уровне редукции. Применяя еще раз операцию 1 над вершиной C и операции 2 над вершинами {A,E}, получаем

A E

В этой записи все вершины изолированы на третом уровне редукции. □

Будем говорить об изолированных и связанных вершинах, подразумевая, где это возможно, соответствующий уровень редукции.

Легко видеть, что если данная схема базы данных является ациклической, то выполнено одно из следующих утверждений:

а) выход алгоритма редукции (т.е. пустая запись) получается путем применения только операции 1 алгоритма редукции. В этом случае все вершины гиперграфа (все атрибуты схемы) являются изолированными и никакой атрибут не участвует хотя бы два раза в отдельных реляционных схемах, т.е. для всех ребер e гиперграфа H, $con(e) = \varnothing$. Такие базы данных содержат только семантически несвязанные данные [WoM77] и [Ang81], сгруппированные в отдельные несвязанные отношения, не имеющие общих атрибутов. Таких схем баз данных мы рассматривать не будем.

б) выход алгоритма редукции получается путем применения как операцию 1 алгоритма редукции, так и операцию 2 этого алгоритма. В этом случае схема базы данных содержит реляционные схемы, которые имеют общие атрибуты между собой и поэтому являются семантически связанными [WoM77] и [Ang81].

ОПРЕДЕЛЕНИЕ 2.9. Пусть $H = \{e_1, e_2, \ldots, e_p\}$ - данный гиперграф. H будем называть <u>связанным гиперграфом</u>, если для каждой пары ребер $(e_i, e_j)$, $1 \le i \le p$, $1 \le j \le p$, $i \ne j$, существует цепочка различных ребер $e_i - e'_1, e'_2, \ldots, e'_s - e_j$ такие, что для $1 \le r \le s$

1) $e'_r \neq e_1$, если $r \neq 1$;

2) $e'_r \neq e_j$, если $r \neq s$;

3) $e'_r \cap e'_{r+1} \neq \varnothing$ если $r \neq s$;

4) $e'_r \in H$. □

ЛЕММА 2.1. Если H - связанный гиперграф, то на всех уровнях алгоритма редукции над H получается запись, соответствующая некоторому связанному гиперграфу.

ДОКАЗАТЕЛЬСТВО. Пусть n - соответствующий уровень редукции. Лемму докажем с помощью индукции по отношению к n.

n-1. Покажем, что полученная запись на первом уровне редукции соответствует некоторому связанному гиперграфу H'. Пусть $H=\{e_1,e_2,\ldots,e_p\}$, а $H'=\{e'_1,e'_2,\ldots,e'_p\}$. Докажем, что H' - связанный гиперграф. На первом уровне редукции имеем $e'_i$-con$(e_i)$, $1 \leq i \leq p-1$. Строка $e_p \in H$ зачеркнута с помощью операции 2 алгоритма редукции - она поглощена некоторой строкой $e_r \in H$.

Пусть $e'_i, e'_j$ - ребра из H'. Покажем, что они связанны.

Пусть $e_i \in H$, $e_j \in H$ связанны при помощи цепочки $e_i$-$e_{i1}, e_{i2}, \ldots, e_{is}$-$e_j$. Если $e_{ik} \neq e_p$ для $k=1,2,\ldots,s$, то тогда $e'_i$ и $e'_j$ связанны при помощи цепочки $e'_i$-$e'_{i1}, e'_{i2}, \ldots, e'_{is}$-$e'_j$. Если $e_{ik}$-$e_p$ для $k=m$, то тогда $e'_i$ и $e'_j$ связанны при помощи цепочки $e'_i$-$e'_{i1}, \ldots, e'_{i\,m-1}, e'_r, e'_{i\,m+1}, \ldots, e'_{is}$-$e'_j$.

Следовательно, полученная на первом уровне редукции запись

соответствует некоторому связанному гиперграфу.

Допустим, что на n-том уравне редукции полученная запись соответствует некоторому связанному гиперграфу.

Видно, что на n+1-ом уровне редукции (анналогично, как на первом уравне редукции) получается запись соответствующая некоторому связанному гиперграфу. □

ЛЕММА 2.2. Гиперграф H является ациклическим тогда и только тогда, когда на всех уровнях редукции гиперграфа H получается запись, соответствующая некоторому ациклическому гиперграфу.

ДОКАЗАТЕЛЬСТВО. Следует непосредственно из теоремы 2.1 и из определения 2.3. □

Отметим, что в процессе редукции ациклического гиперграфа пустую запись можно получить только с помощью операции 1. Последная строка, над которой в этом случае применяется операция 1, поглотила все остальные строки на последном уровне редукции данного гиперграфа.

ПРИМЕР 2.7. Покажем, что поглощающая строка на последнем уровне редукции ациклического гиперграфа зависит от порядка применения операции 2 алгоритма редукции.

Рассмотрим гиперграф

| | | |
|---|---|---|
| A | B | |
| | B | C |
| | | C D |

Если применить операцию 2 над парой {A,B} и {B,C}, где con{A,B} ⊂ con{B,C}, получим в редуцированной на первом уровне записи строку {B,C}; затем если применить операцию 2 над парой {B,C} и {C,D}, где con({B,C}) ⊂ con({C,D}), то тогда поглощающая строка этого гиперграфа будет {C,D}. Но если применить операцию

2 на первом уровне редукции над парой $con\{C,D\} \subset con\{B,C\}$, то тогда поглощающая строка на втором уровне редукции может быть $\{B,C\}$. □

Лемма 2.2 и пример 2.7 показывают, что поглощающая строка всегда существует для ациклических гиперграфов, но какая она конкретно - это зависит от выбора порядка применения операции 2 алгоритма редукции гиперграфов.

ОПРЕДЕЛЕНИЕ 2.10. [Fa83a] Гиперграф H называется <u>A-ациклическим</u>, если алгоритм редукции заканчивает работу с пустой записью над этим гиперграфом. □

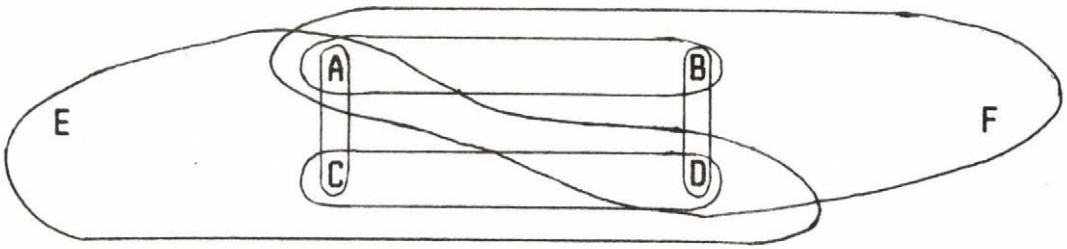
ОПРЕДЕЛЕНИЕ 2.11. Пусть $H=\{e_1,e_2,\ldots,e_s\}$ - A-ациклический гиперграф. <u>Циклической компонентой</u> в H будем называть такую совокупность ребер H', $H' \subset H$, для которой алгоритм редукции не заканчивает работу с пустой записью над входом H'. □

Как уже видно из леммы 2.2, для каждого A-ациклического гиперграфа H существует ребро, которое поглощает все остальные ребра при применении алгоритма редукции над H.

ОПРЕДЕЛЕНИЕ 2.12. Пусть $H=\{e_1,e_2,\ldots,e_s\}$ - гиперграф. Пусть его ребро $e_i$ поглощает ребро $e_j$ путем одного применения операции 2 алгоритма редукции. Тогда $e_i$ будем называть <u>непосредственно поглощающим</u> для ребра $e_j$. □

ПРИМЕР 2.8. Лемма 2.2 показывает, что для циклических компонентов в A-ациклическом гиперграфе всегда существует

поглощающее ребро на некотором уровне редукции. Но как видно из следующей схемы базы данных, непосредственное поглощающее ребро может не быть единственным для циклических компонентов в А-ациклических гиперграфах.




Этот гиперграф - А-ациклический. Здесь циклическая компонента состоит из ребер {A,C}, {A,B}, {B,D} и {C,D}. Ребро {A,C,D,E} - непосредственно поглощающее ребро для {A,C} и {C,D}, а ребро {A,B,D,F} - непосредственно поглощающее ребро для {A,B} и {B,D}. □

ОПРЕДЕЛЕНИЕ 2.13. [Fag83a] Пусть H - А-ациклический гиперграф, который не содержит циклических компонент. Тогда будем называть H <u>В-ациклическим</u> гиперграфом. Все гиперграфы, которые не являются В-ациклическими, будем называть <u>В-циклическими</u>. □

ТЕОРЕМА 2.4. [Fag83] Гиперграф H является В-циклическим тогда и только тогда, когда в нем существует цепочка

$(S_1, x_1, S_2, x_2, \ldots, S_m, x_m, S_{m+1})$

такая, что

1) $x_1, x_2, \ldots, x_m$ - разные вершины гиперграфа;

2) $S_1, S_2, \ldots, S_m$ - разные ребра гиперграфа, а $S_{m+1} = S_1$;

3) $m \geq 3$, т.е. цепочка содержит по крайней мере три ребра;

4) $x_l \in S_l \cap S_{l+1}$ $(1 \leq l \leq m)$ и никакому другому $S_j$. □

ЛЕММА 2.3. Алгоритм редукции заканчивает работу с пустой записью над каждым подмножеством данного гиперграфа H тогда и только тогда, когда H является B-ациклическим.

ДОКАЗАТЕЛЬСТВО. а) Достаточность. Пусть H является B-ациклическим, т.е. по теореме 2.4 в нем нет цепочки циклически связанных ребер. Следовательно, в каждом $H', H' \subset H$, тоже нет такой цепочки и по теореме 2.4 каждое множество $H'$ является тоже B-ациклическим. Тогда алгоритм редукции заканчивает работу с пустой записью над $H'$.

б) Необходимость. Пусть алгоритм 2.1 заканчивает работу с пустой записью над каждым подмножеством данного гиперграфа H. Допустим, что H не является B-ациклическим. Тогда по теореме 2.4 в нем существует цепочка $(S_1, x_1, S_2, x_2, \ldots, S_m, x_m, S_{m+1})$

Покажем, что над этой цепочкой алгоритм редукции не может закончить работу с пустой записью.

Применяем операцию 1 над изолированными вершинами из $S_1, S_2, \ldots, S_m$. В полученную запись участвуют m строки:

$$\text{con}(S_1) = X_m \cup X_1$$

$$\text{con}(S_2) = X_1 \cup X_2$$

$$- - -$$

$$\text{con}(S_m) = X_{m-1} \cup X_m$$

Так как $X_I \neq X_J$ при $I \neq J$, $1 \leq I \leq m$, $1 \leq J \leq m$, то невозможно применить операцию 2 алгоритма редукции над полученной записью. Видно, что невозможно применять и операцию 1 алгоритма редукции - так как полученная запись состоит только из связанных вершин.

Следовательно, над цепочки $(S_1, x_1, S_2, x_2, \ldots, S_m, x_m, S_{m+1})$ алгоритм редукции не может закончить работу с пустой записью, что является противоречием. □

# Г Л А В А  Т Р Е Т А Я

# ПОСТРОЕНИЕ ТАБЛИЦ КОНЪЮНКТИВНЫХ ЗАПРОСОВ НА QUEL

Как было уже отмечено в параграфе 1.5, в [ASU79] конструирован алгоритм построения таблиц только для данных SPJ-выражений. Очевидно, однако, что данному SPJ-выражению соответствует класс эквивалентных таблиц (некоторая из них - с минимальным числом строк; минимальная таблица одинствена с точностью переименования символов). В общем случае мы не знаем как нужно построить таблицу по заданному конъюнктивному запросу. По этой причине весьма возможно, что нам понадобится строить таблицы, числом строк которых больше, чем необходимо, а затем найти лишные строки в процессе оптимизации таблиц. В этой главе предлагается алгоритм построения таблиц, содержащих как можно меньше строк.

В 3.1 рассмотрены некоторые особенности понятия таблицы - а именно, возможное присутствие "лишних" строк. Введено формальное определение понятия лишней строки в данной таблице. В 3.2 и 3.3 дан алгоритм построения таблиц для конъюнктивных запросов на языке QUEL. Особенностью этих таблиц является то, что они содержат минимальное возможное число строк (этот факт будет доказан в гл. 4). В 3.2 введено графическое представление запросов на языке QUEL, а в 3.3 показан алгоритм построения таблиц по данному графическому представлению. Так как в 3.2 и 3.3 изложен способ сопоставления таблиц данному конъюнктивному

запросу, встает вопрос можно ли описанным способом сопоставить каждому конъюнктивному запросу на QUEL некоторую таблицу. В 3.4 дан ответ на этот вопрос. Показано, что существуют такие конъюнктивные запросы, которым вообще нельзя сопоставить некоторую таблицу. Поэтому средства и методы их оптимизации не существуют; в практических применениях их нужно обрабатывать в виде, заданном пользователем. Также показано, что для всех остальных конъюнктивных запросов существует таблица запроса, которую можно построить описанным в 3.2 и 3.3 способом.

## 3.1. Сущность понятия таблицы обуславливает трудности ее оптимизации

Представление конъюнктивных запросов с помощью таблиц запросов не всегда позволяет нам найти более эффективный алгоритм для проверки эквивалентности запросов. Этот факт связан со самой сущностью понятия таблицы запроса, как будет показано на следующем примере.

ПРИМЕР 3.1. Рассмотрим таблицы $T_1$ и $T_2$:

$T_1$: $a_1$———— и $T_2$: $a_1$————

| | | |
|---|---|---|
| $a_1$ | $b_1$ | (R) |

| | | |
|---|---|---|
| $a_1$ | $b_1$ | (R) |
| $b_2$ | $b_3$ | (R) |
| $b_4$ | $b_5$ | (R) |

Сразу видно, что таблицы $T_1$ и $T_2$ - сильно эквивалентны, так как существуют содержащие отображения $h_1$: $T_1 \to T_2$ и $h_2$: $T_2 \to T_1$; при этом $h_1$ и $h_2$ сохраняют маркеры всех строк. Строки $\langle b_2\ b_3 \rangle$ и

$<b_4\ b_5>$ не добавляют новую информацию к информации, вытекающей из строки $<a_1\ b_1>$ в таблице $T_2$. С помощью содержащего отображения $h_2$: $h_2(a_1)-a_1$, $h_2(b_1)-b_1$, $h_2(b_2)-a_1$, $h_2(b_3)-b_1$, $h_2(b_4)-a_1$, $h_2(b_5)-b_1$ таблицу $T_2$ можно "сжать" в таблицу $T_1$; таким образом, строки $<b_2\ b_3>$ и $<b_4\ b_5>$ можно "сжать" в строку $<a_1\ b_1>$. □

Вообще алгоритм для проверки эквивалентности двух данных таблиц является NP-полным и в связи с тем, что мы должны учитывать возможное присутствие "лишних" строк в некоторой заданной таблице.

Этот пример позволяет нам ввести следующее определение:

ОПРЕДЕЛЕНИЕ 3.1. Пусть T и T′ - эквивалентные таблицы, причем T′ содержит меньше строк, чем T. Пусть T′ получается из T в процессе оптимизации T. (Следовательно строки T′ являются подмножеством строк T). Пусть содержащие отображения между T и T′ соответственно $h:T\to T'$ и $h_1:T'\to T$. Существуют по меньшей мере две строки $r_1$ и $r_2$ таблицы T так, что строка $r_1$ принадлежит таблицы T′ и $h(r_1)-r_1$, $h(r_2)-r_1$. Тогда строку $r_2$ будем называть лишней строкой для таблицы T. □

ПРИМЕР 3.2. Рассмотрим таблицу $T_2$ из примера 3.1 и таблицу $T_2'$:

$T_2$: $a_1$———— и $T_2'$: $a_1$————

| | | | | | |
|---|---|---|---|---|---|
| $a_1$ | $b_1$ | (R) | $a_1$ | $b_1$ | (R) |
| $b_2$ | $b_3$ | (R) | $b_2$ | $b_3$ | (R) |
| $b_4$ | $b_5$ | (R) | | | |

Очевидно, что $T_2$ и $T_2'$ - сильно эквивалентны. Таблица $T_2'$ получается из таблицы $T_2$ в процессе ее оптимизации и строки таблицы $T_2'$ являются подмножеством строк таблицы $T_2$. Рассмотрим следующие содержащие изображения между двумя таблицами: $h: T_2 \to T_2'$ и $h_1$: $T_2' \to T_2$, такие, что $h(a_1)=a_1$, $h(b_1)=b_1$, $h(b_2)=b_2$, $h(b_3)=b_3$, $h(b_4)=a_1$, $h(b_5)=b_1$, $h_1(a_1)=a_1$, $h_1(b_1)=b_1$, $h_1(b_2)=b_2$, $h_1(b_3)=b_3$.

Отображения $h_1$ и $h_2$ сохраняют маркеры всех строк.

Тогда $h(\langle a_1 \quad b_1\rangle)=\langle a_1 \quad b_1\rangle$, $h(\langle b_4 \quad b_5\rangle)=\langle a_1 \quad b_1\rangle$ и следовательно, строка $\langle b_4 \quad b_5\rangle$ является лишней строкой для таблицы $T_2$. □

## 3.2. Таблицы доступа и графическое представление конъюнктивных запросов

В этой части мы введем новый способ представления конъюнктивных запросов из $K1$, который будем использовать при изучении минимальных запросов. Конъюнктивные запросы будут представлены как структуру в терминах элементов схемы базы данных. Схема базы данных будет представлена с помощью так называемой "таблицы доступа".

ОПРЕДЕЛЕНИЕ 3.2. Пусть дано множество атрибутов $U=\{A_1,A_2,\ldots,A_k\}$ и множество реляционных схем $R=\{R_1,R_2,\ldots,R_n\}$ над атрибутами $U$. Таблицу с n строками и k столбами будем называть <u>таблицей доступа</u>, если:

- каждому атрибуту из $U$ сопоставлен точно один столбец таблицы и наоборот, каждому столбцу таблицы сопоставлен точно один атрибут из $U$;
- каждой реляционной схеме из $R$ сопоставлена точно одна строка таблицы и наоборот, каждой строке таблицы сопоставлена точно одна реляционная схема из $R$;
- значения элементов $a_{ij}$ таблицы следующие:

$$a_{ij}=\begin{cases}1 & \text{- если атрибут, соответствующий J-тому столбцу, участвует в реляционной схеме, соответствующей I-той строке.}\\ \text{пробел} & \text{- в противном случае.}\end{cases}$$ □

ПРИМЕР 3.3. Рассмотрим множество атрибутов:

$U=\{$ИМЯ,ДОЛЖНОСТЬ,ЗАРПЛАТА,ОТДЕЛ#,ИМЯ-ОТДЕЛА$\}$ и пусть

$R=\{$СЛУЖИТЕЛЬ,ОТДЕЛ $\}$, где

СЛУЖИТЕЛЬ$=\{$ИМЯ,ДОЛЖНОСТЬ,ЗАРПЛАТА,ОТДЕЛ#$\}$,

ОТДЕЛ$=\{$ОТДЕЛ#,ИМЯ-ОТДЕЛА$\}$.

Тогда схеме базы данных $R$ сопоставляется следующая таблица доступа:

| | <u>ИМЯ</u> | <u>ДОЛЖНОСТЬ</u> | <u>ЗАРПЛАТА</u> | <u>ОТДЕЛ#</u> | <u>ИМЯ-ОТДЕЛА</u> |
|---|---|---|---|---|---|
| СЛУЖИТЕЛЬ | 1 | 1 | 1 | 1 | |
| ОТДЕЛ | | | | 1 | 1 |

□

Как видно, таблица доступа описывает способ, которым

атрибуты из U участвуют в отдельных реляционных схемах. Она представляет собой "прообраз" (макет) всех таблиц запросов к данной схеме базы данных. Таблица доступа напоминает начальное представление схемы базы данных, над которым применяется алгоритм редукции 2.1.

Ниже мы будем называть столбцы и строки данной таблицы доступа "столбцом $A_j$" и "строкой $R_i$", отмечая таким образом соответствующие им атрибуты и реляционные схемы.

Следующие определения вводят представления конъюнктивного запроса, заданного к фиксированной схеме базы данных.

ОПРЕДЕЛЕНИЕ 3.3. Пусть q - конъюнктивный запрос на языке QUEL, записанный в виде (1.1). В квалификации запроса q содержатся условия, связывающие попарно имена реляционных схем - точнее, условия

$(R_{k1}.A_{k1} \ominus R_{k2}.A_{k1})$ AND $(R_{k3}.A_{k3} \ominus R_{k4}.A_{k3})$ AND...

AND $(R_{kt}.A_{kt} \ominus R_{k(t+1)}.A_{kt})$

Эти условия мы будем называть <u>путем доступа</u> для запроса q. □

ОПРЕДЕЛЕНИЕ 3.4. Пусть q - конъюнктивный запрос на языке QUEL, с путем доступа

$(R_{k1}.A_{k1} \ominus R_{k2}.A_{k1})$ AND $(R_{k3}.A_{k3} \ominus R_{k4}.A_{k3})$ AND... (3.1)

AND $(R_{kt}.A_{kt} \ominus R_{k(t+1)}.A_{kt})$ .

Если $\ominus$ в (3.1) везде является "=", определим <u>графическое представление пути доступа</u>

$(R_{k1}.A_{k1}=R_{k2}.A_{k1})$ AND $(R_{k3}.A_{k3}=R_{k4}.A_{k3})$ AND... (3.2)

AND $(R_{kt}.A_{kt}=R_{k(t+1)}.A_{kt})$ .

Графическое представление этого пути доступа будем строит в

виде графа G= (N,E) в соответствующей таблице доступа следующим образом:

- для каждого условия пути доступа $R_{ks}.A_{ks}=R_{k(s+1)}.A_{ks}$, s-1,3,...,t, элементы в таблице доступа, которые находятся в столбце $A_{ks}$, строке $R_{ks}$ и в столбце $A_{ks}$, строке $R_{k(s+1)}$, будут вершинами графа, связанными с ребром ( $(R_{ks},A_{ks})$ , $(R_{k(s+1)},A_{ks})$ ). Так как путь доступа (3.2) содержит (t+1)/2 таких условий, мы получаем (t+1)/2 ребер, связывающих соответственно t+1 вершин графа;

- для каждой реляционной схемы $R_i$ из R, составляем список ее появлений в объектном списке запроса q, в условиях задающих сравнения с константами в квалификации запроса q и в пути доступа (3.2). Пусть этот список разных появлений будет:

$$R_i.A_{i1},\ R_i.A_{i2},\ \ldots,\ R_i.A_{ip}.$$

Упорядочиваем элементы списка в соответствии с порядком атрибутов в столбцах таблицы доступа:

$$R_i.A_{i1}',\ R_i.A_{i2}',\ \ldots,\ R_i.A_{ip}'.$$

Каждый элемент этого списка является элементом также таблицы доступа со значением 1; каждому такому элементу мы сопоставляем вершину графического представления пути доступа (3.2). Эти вершины мы связываем попарно с ребрами:

$$((R_i.A_{i1}'),(R_i.A_{i2}')),\ ((R_i.A_{i2}'),(R_i.A_{i3}')),\ \ldots,$$

$$((R_i.A_{i(p-1)}'),(R_i.A_{ip}')).$$

Полученная совокупность вершин и ребер в рамках соответствующей таблицы доступа будем называть графическим представлением пути доступа (3.2).

Видно, что вершинами графического представления пути доступа

(3.2) выступают только те элементы таблицы доступа, значения которых равняются 1. □

ОПРЕДЕЛЕНИЕ 3.5. Пусть q - конъюнктивный запрос на языке QUEL, записанный в виде (1.3):

$$q: \text{RETRIEVE } (R_{l1}.A_{l1}, R_{l2}.A_{l2}, \ldots, R_{ls}.A_{ls})$$

$$\text{WHERE } (R_{j1}.A_{j1} = c_1) \text{ AND } (R_{j2}.A_{j2} = c_2) \text{ AND} \ldots \text{AND } (R_{jp}.A_{jp} = c_p)$$

$$\text{AND } (R_{k1}.A_{k1} = R_{k2}.A_{k1}) \text{ AND } (R_{k3}.A_{k3} = R_{k4}.A_{k3}) \text{ AND} \ldots$$

$$\text{AND } (R_{kt}.A_{kt} = R_{k(t+1)}.A_{kt}) \ .$$

Вводим <u>представление для запроса q в виде упорядоченной четверки</u>:

$q = (T_L, T_{con}, T_{rr}, \Sigma)$, где

$T_L = \{R_{l1}.A_{l1}, R_{l2}.A_{l2}, \ldots, R_{ls}.A_{ls}\}$; $T_L$ содержит элементы объектного списка запроса q;

$T_{con} = \{ R_{j1}.A_{j1} = c_1, \ R_{j2}.A_{j2} = c_2, \ \ldots, \ R_{jp}.A_{jp} = c_p \}$; $T_{con}$ содержит все условия квалификации запроса, в которых участвует конкретная константа;

$T_{rr}$ - графическое представление пути доступа

$$(R_{k1}.A_{k1} = R_{k2}.A_{k1}) \text{ AND } (R_{k3}.A_{k3} = R_{k4}.A_{k3}) \text{ AND} \ldots$$

$$\text{AND } (R_{kt}.A_{kt} = R_{k(t+1)}.A_{kt});$$

$\Sigma = \{R_{ks}.A_{ks} \ \theta \ R_{k(s+1)}.A_{ks} \mid \theta$ не является сравнением "=" для $s = 1, 3, \ldots, t\}$.

Так как равенство под формой сравнения присутствует в графическом представлении $T_{rr}$, в $\Sigma$ включены все отношения, которые не являются равенством. $\Sigma$ напоминает нам список ограничений, который можно добавить к каждой таблице запроса.

Введенное таким образом представление запроса q в виде упорядоченной четверки будем называть для краткости графическим представлением запроса q. □

ПРИМЕР 3.4. Для схемы в примере 3.3 рассмотрим запрос

$q_1$ : Найти имя служителя, который является начальником отдела игрушек.

```
RETRIEVE (СЛУЖИТЕЛЬ.ИМЯ)
   WHERE (СЛУЖИТЕЛЬ.ДОЛЖНОСТЬ = "начальник")
     AND (СЛУЖИТЕЛЬ.ОТДЕЛ# = ОТДЕЛ.ОТДЕЛ#)
     AND (ОТДЕЛ.ИМЯ-ОТДЕЛА = "игрушки") .
```

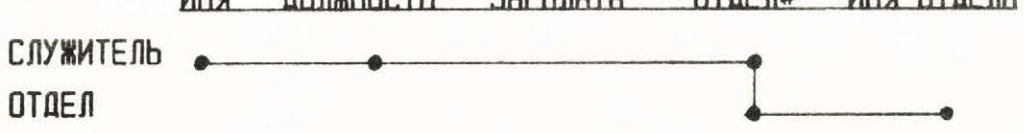
Представим запрос $q_1$ в виде упорядоченной четверки

$q_1 = (T_L, T_{con}, T_{rr}, \Sigma)$ следующим образом:

$T_L$ = {СЛУЖИТЕЛЬ.ИМЯ} ,

$T_{con}$ = {ОТДЕЛ.ИМЯ ОТДЕЛА = "игрушки",
СЛУЖИТЕЛЬ.ДОЛЖНОСТЬ= "начальник"},

$T_{rr}$ следующее графическое представление:




$\Sigma = \varnothing$ . □

ПРИМЕР 3.5. Для базы данных в примере 3.3 рассмотрим также запрос, содержащий больше чем один RANGE-оператор для некоторой реляционной схемы:

$q_2$ : Найти имена и должности служителей, зарплата которых превышает зарплату некоторого начальника.

RANGE OF $a_1$ IS СЛУЖИТЕЛЬ

RANGE OF $a_2$ IS СЛУЖИТЕЛЬ

RETRIEVE ($a_1$.ИМЯ, $a_1$.ДОЛЖНОСТЬ)

WHERE ($a_2$.ДОЛЖНОСТЬ = "начальник")

AND ($a_1$.ЗАРПЛАТА > $a_2$.ЗАРПЛАТА) .

Запросу $q_2$ соответствует следующая запись в виде (1.1) как конъюнктивного запроса:

$$\{a_1 a_2 \mid (\exists b_1)(\exists b_2)(\exists b_3)(\exists b_4)(\exists b_5) : \text{СЛУЖИТЕЛЬ}(a_1, a_2, b_1, b_2) \ \&$$
$$\text{СЛУЖИТЕЛЬ}(b_3, \text{"начальник"}, b_4, b_5) \ \& \ (b_1 > b_4) \}.$$

С помощью этой записи строим таблицу

| ИМЯ | ДОЛЖНОСТЬ | ЗАРПЛАТА | ОТДЕЛ# | ИМЯ-ОТДЕЛА | |
|---|---|---|---|---|---|
| $a_1$ | $a_2$ | | | | |
| $a_1$ | $a_2$ | $b_1$ | $b_2$ | | (СЛУЖИТЕЛЬ) |
| $b_3$ | начальник | $b_4$ | $b_5$ | | (СЛУЖИТЕЛЬ') |
| | | $b_1 > b_4$ | | | |

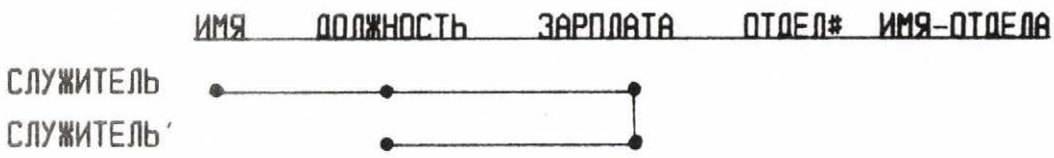
Запросу $q_2$ можно также сопоставить представление в виде упорядоченной четверки, записав в таблицу доступа строку для реляционной схемы СЛУЖИТЕЛЬ во второй раз. Тогда

$q_2 = (T_L, T_{con}, T_{rr}, \Sigma)$ , где

$T_L$ = {СЛУЖИТЕЛЬ.ИМЯ, СЛУЖИТЕЛЬ.ДОЛЖНОСТЬ},

$T_{con}$ = {СЛУЖИТЕЛЬ'.ДОЛЖНОСТЬ - "начальник"} ,

$T_{rr}$:




$\Sigma$ = {СЛУЖИТЕЛЬ.ЗАРПЛАТА > СЛУЖИТЕЛЬ'.ЗАРПЛАТА} .

Этот пример показывает, что графическое представление конъюнктивных запросов, введенное с помощью определения 3.5, можно расширить для конъюнктивных запросов, у которых больше, чем один RANGE-оператор для некоторой реляционной схемы. □

Отметим, что по заданному графическому представлению конъюнктивного запроса сразу можно восстановить запись запроса в виде (1.1).

В определении 3.4 вводится графическое представление данного пути доступа:

$(R_{k1}.A_{k1}=R_{k2}.A_{k1})$ AND $(R_{k3}.A_{k3}=R_{k4}.A_{k3})$ AND... (3.2)

AND $(R_{kt}.A_{kt}=R_{k(t+1)}.A_{kt})$ .

В строках соответствующей таблицы доступа вершины графического представления упорядочены в соответствии со столбцами таблицы доступа. Если $q=(T_L, T_{con}, T_{rr}, \varnothing)$, то элементы столбцов $T_{rr}$ тоже могут быть упорядочены в соответствии с порядком строк данной таблицы доступа.

ОПРЕДЕЛЕНИЕ 3.6. Пусть $q=(T_L, T_{con}, T_{rr}, \varnothing)$. Определим <u>упорядоченное графическое представление</u> $T^1_{rr}$, эквивалентно представлению $T_{rr}$:

Графическое представление пути доступа q построено в виде графа G= (N,E) в соответствующей таблице доступа.

Для каждого атрибута A из U, составляем список его появлений в объектном списке запроса q, в условиях задающих сравнения с константами в квалификации запроса q и в пути доступа (3.2). Пусть этот список разных появлений будет:

$$R_{I1}.A,\ R_{I2}.A,\ \ldots,\ R_{Iv}.A.$$

Упорядочиваем элементы списка в соответствии с порядком реляционных схем в строках таблицы доступа:

$$R_{I1}'.A,\ R_{I2}'.A,\ \ldots,\ R_{Iv}'.A.$$

Каждому такому элементу сопоставлена вершина графического представления пути доступа (3.2).

Удаляем существующие ребра между этими вершинами и связываем их попарно с ребрами:

$$((R_{I1}'.A),(R_{I2}'.A)),\ ((R_{I2}'.A),(R_{I3}'.A)),\ \ldots,$$

$$((R_{I(v-1)}'.A),(R_{Iv}'.A)).$$

Полученная совокупность вершин и ребер в рамках соответствующей таблицы доступа будем называть упорядоченным графическим представлением пути доступа (3.2). □

ПРИМЕР 3.6. Рассмотрим базу данных из примера 1.4:

ЧАСТЬ (ЧИМЯ,ЧНОМЕР,ЦЕНА)
ПОСТАВЩИК (ПИМЯ,ПНОМЕР,ПАДРЕС,ПГОРОД)
КЛИЕНТ (КИМЯ,КНОМЕР,КАДРЕС,КГОРОД)
ПОСТАВКА (ЧНОМЕР,ПНОМЕР,КНОМЕР,КОЛИЧЕСТВО)
ОБЯЗАННОСТЬ (ЧНОМЕР,ПНОМЕР).

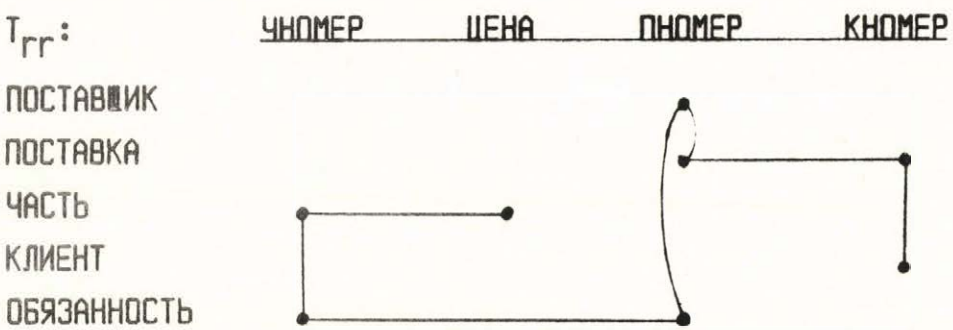
Пусть $q_3$ - запрос

$q_3$: RETRIEVE (ОБЯЗАННОСТЬ.ПНОМЕР)

WHERE (ЧАСТЬ.ЦЕНА='200')
AND (ОБЯЗАННОСТЬ.ЧНОМЕР=ЧАСТЬ.ЧНОМЕР)
AND (ОБЯЗАННОСТЬ.ПНОМЕР=ПОСТАВЩИК.ПНОМЕР)
AND (ПОСТАВЩИК.ПНОМЕР=ПОСТАВКА.ПНОМЕР)
AND (ПОСТАВКА.КНОМЕР=КЛИЕНТ.КНОМЕР).

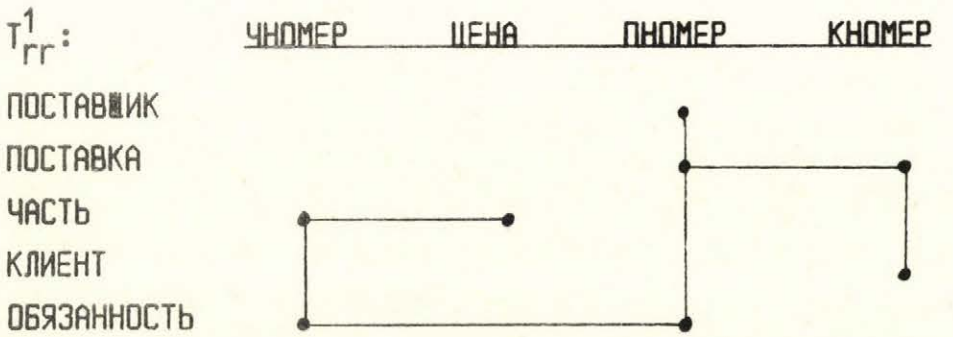
Если строки таблицы доступа упорядочены в соответствии со списком {ПОСТАВЩИК, ПОСТАВКА, ЧАСТЬ, КЛИЕНТ, ОБЯЗАННОСТЬ}, то тогда графическое представление запроса $q_3$ выглядит следующим образом:

$T_{rr}$:




Это графическое представление может быть упорядочено в столбце для атрибута ПНОМЕР следующим образом:

$T^1_{rr}$:




□

Видно, что семантика отношения "=" (точнее, его транзитивность) позволяет нам менять синтаксис данного запроса q, не меняя "смысл" q. Мы воспользуемся этим фактом в гл. 5, в которой на упорядоченном графическом представлении пути доступа показан алгоритм проверки эквивалентности запросов к реляционной базе данных.

Дальше, если $q=(T_L,T_{con},T_{rr},\varnothing)$, мы будем исспользовать упорядоченное графическое представление пути доступа q.

В конце параграфа отметим, что разные графические представления запросов уже вводились в других целях - например в алгоритме декомпозиции запросов пользователя на языке QUEL [WoY76]; кроме того, так называемые древовидные запросы (tree queries) [GoS82a] тоже имеют графическое представление.

## 3.3 Графическое представление и таблицы конъюнктивных запросов

Рассмотрим следующий алгоритм:

АЛГОРИТМ 3.1. Построение таблицы по данному графическому представлению конъюнктивного запроса.

ВХОД: Графическое представление конъюнктивного запроса q на языке QUEL с одним RANGE-оператором для каждой реляционной схемы - $q=(T_L,T_{con},T_{rr},\Sigma)$.

ВЫХОД: Таблица, соответствующая входному графическому представлению запроса.

МЕТОД: Пусть входное графическое представление расположено в соответствующей таблице доступа в строках $R_1,R_2,\ldots,R_m$. Выходная таблица будет содержать m строк с маркерами соответственно $R_1$, $R_2,\ldots,R_m$ и столбцов соответствующих

атрибутах из U, но заполняться будут только столбцы $\cup_{i=1}^{m} R_i$.

1. Строим резюме таблицы, используя множество

$T_L = \{R_{i1}.A_{i1}, \ldots, R_{is}.A_{is}\}$ - записываем последовательно свободные переменные $a_1, a_2, \ldots, a_s$ в столбцах $A_{i1}$ из $R_{i1}$, $A_{i2}$ из $R_{i2}, \ldots, A_{is}$ из $R_{is}$.

2. Рассматриваем множество

$T_{con} = \{ R_{jk}.A_{jk} \theta c_k \mid k=1,2,\ldots,p \}$. Для каждого условия $R_{jk}.A_{jk} \theta c_k$, $k=1,\ldots,p$ делаем:

а) если $\theta$ является "=", вносим константу $c_k$ в столбец $A_{jk}$, в строку с маркером $R_{jk}$;

б) если $\theta$ не является "=", в столбец $A_{jk}$ в строку $R_{jk}$ таблицы вносим новую связанную переменную b и к ограничениям таблицы добавляем $b \theta c_k$;

3. Для каждой строки $R_k$ элемента $T_{rr}$ графического представления, $k=1,2,\ldots,m$ оформляем строку в таблице следующим образом: пусть схема $R_k = \{A_{k1}, A_{k2}, \ldots, A_{kl}\}$; тогда для $1 \le l \le l$ делаем:

а) если $R_k.A_{kl} \in T_L - T_{con}$, ставим в строку $R_k$ таблицы соответствующую свободную переменную a столбца $A_{kl}$ резюме;

б) если $R_k.A_{kl} \in T_{con} - T_L$, в таблицу не вносится символ - там уже поставлена соответствующая константа или некоторая связанная переменная b;

в) если $R_k.A_{kl} \in T_L \cap T_{con}$, тогда $R_k.A_{kl} \theta c_k$. Если $\theta$ является "=", в резюме таблицы ставим константу $c_k$, записывая ее в

столбец $A_{kl}$ вместо уже внесенной свободной переменной a;

г) если $R_k.A_{kl} \in T_L \cap T_{con}$, тогда $R_k.A_{kl} \theta c_k$. Если $\theta$ не является "=", вместо записанной уже в таблице связанной переменной b записываем соответствующую константу $c_k$ в строку $R_k$, в столбец $A_{kl}$; стираем со списка ограничений таблицы ограничение $b \theta c_k$ и добавляем новое ограничение $a \theta c_k$, где a - свободная переменная столбца $A_{kl}$ резюме таблицы;

д) если $R_k.A_{kl}$ не участвует в $T_L$ и $T_{con}$, вносим в строку $R_k$ таблицы, в столбец $A_{kl}$ новую связанную переменную b.

4. Заполнив символами все строки $R_k$ таблицы, $1 \leq k \leq m$, рассмотрим ребра графа $T_{rr}$ графического представления, связывающие вершины различных строк (но в одном и том же столбце). Пусть такое ребро связывает например вершину $R_k.A_{kl}$ с вершиной $R_j.A_{kl}$. Если в $\Sigma$ было условие $R_k.A_{kl} \theta R_j.A_{kl}$ и $\theta$ не является "=", то к ограничениям таблицы добавляем $s_1 \theta s_2$, где $s_1$ и $s_2$ - соответственно символы, поставленные до этого момента в строку $R_k$, столбец $A_{kl}$ и в строку $R_j$, столбец $A_{kl}$ таблицы. Если $R_k.A_{kl} = R_j.A_{kl}$ (т.е., в $\Sigma$ запроса q подобного ограничения не существует), тогда символы $s_1$ и $s_2$ будут отождествляться по следующим правилам:

а) если $s_1 - a_1, s_2 - a_2$ - таблица $T = \varnothing$;

б) если $s_1 - a$, $s_2 - b$ - вносим вместо $s_2$ переменную a;

в) если $s_1 - a$, $s_2 - c$ - вносим вместо $s_1$ константу c;

г) если $s_1 - b$, $s_2 - a$ - вносим вместо $s_1$ переменную a;

д) если $s_1$-$b_p$, $s_2$-$b_q$ - вносим вместо $s_1$ и $s_2$ связанную переменную $b_r$, где $r=\min(p,q)$;

е) если $s_1$-b, $s_2$-c - вносим вместо $s_1$ константу c;

ж) если $s_1$-c, $s_2$-a - вносим вместо $s_2$ константу c;

з) если $s_1$-c, $s_2$-b - вносим вместо $s_2$ константу c;

и) если $s_1$-$c_1$, $s_2$-$c_2$ - таблица $T=\varnothing$. □

ПРИМЕР 3.6. Для запроса из примера 3.4 строим таблицу:

| ИМЯ | ДОЛЖНОСТЬ | ЗАРПЛАТА | ОТДЕЛ | ИМЯ-ОТДЕЛА |
|---|---|---|---|---|
| $a_1$ | | | | |
| $a_1$ | начальник | $b_1$ | $b_2$ | (СЛУЖИТЕЛЬ) |
| | | | $b_2$ игрушки | (ОТДЕЛ) |

Здесь "начальник" и "игрушки" являются константами соответствующего домена. □

С помощью алгоритма 3.1 строим таблицу, которая обладает следующими свойствами:

1. Столбцы таблицы соответствуют атрибутам универсума U в фиксированном порядке.

2. Символы таблицы - свободные переменные, связанные переменные, константы и пробелы.

3. В резюме можно вносить только свободные переменные, константы и пробелы.

4. Если некоторая свободная переменная появляется в данном столбце резюме, она может появиться в других строках таблицы только в этом столбце.

5. Строки таблицы помечены именами реляционных схем базы данных.

6. В каждой строке заполнени символами только столбцы, соответствующие атрибутам реляционной схемы, имя которой является маркером строки.

7. К таблице можно добавить список неравенств, в котором участвуют символы таблицы.

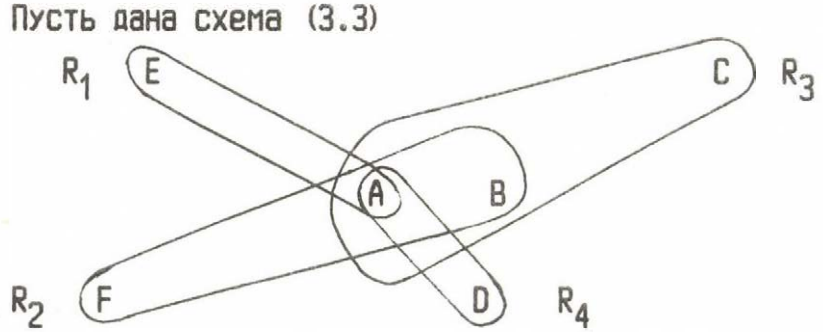
В силу определений в [ASU79] и в [ASU79a], ясно, что построенная с помощью алгоритма 3.1 структура является таблицей некоторого запроса.

Алгоритм 3.1 доказывает следующую лемму:

ЛЕММА 3.1. Пусть q - конъюнктивный запрос на языке QUEL, который содержит наибольше по одному RANGE-оператору для каждой реляционной схемы базы данных. Тогда для q можно построить таблицу T так, чтобы каждая реляционная схема, появляющаяся в записи q на QUEL, является маркером ровно одной строки таблицы T. □

ПРИМЕР 3.7. Покажем, что класс конъынктивных запросов на QUEL, у которых наибольше по одному RANGE-оператору для каждой реляционной схемы, не является подмножеством класса запросов с простыми таблицами.

Пусть дана схема (3.3)




(3.3)

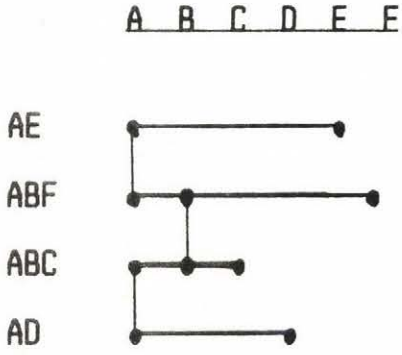
и запрос к этой схеме

RETRIEVE ($R_1$.E)

WHERE ($R_4$.D=$c_1$) AND ($R_3$.C=$c_2$)

AND ($R_1$.A=$R_2$.A) AND ($R_2$.B=$R_3$.B) AND ($R_3$.A=$R_4$.A) .

Для этого запроса составляем соответственно графическое представление пути доступа




и таблицу T:

| A | B | C | D | E | F | |
|---|---|---|---|---|---|---|
| | | | | a | | |
| $b_1$ | | | | a | | (AE) |
| $b_1$ | $b_2$ | | | | $b_3$ | (ABF) |
| $b_4$ | $b_2$ | $c_2$ | | | | (ABC) |
| $b_4$ | | | $c_1$ | | | (AD) |

Как видно из элементов таблицы в столбце A, рассматриваемая таблица не является простой. □

ПРИМЕР 3.8. Покажем, что класс запросов с простыми таблицами не содержится в классе конъюнктивных запросов на QUEL, у которых наибольше по одному RANGE-оператору для каждой реляционной схемы.

Рассмотрим простую таблицу

| A | B | |
|---|---|---|
| $a_1$ | | |
| $a_1$ | $b_1$ | (R) |
| $b_3$ | $b_2$ | (R) |

$b_1 > b_2$

Этой таблице соответствует конъюнктивный запрос

$$\{ a_1 \mid R(a_1,b_1) \ \& \ R(b_3,b_2) \ \& \ (b_1>b_2) \} \ .$$

Как видно, для записи этого запроса на QUEL необходимы два RANGE-оператора для реляционной схемы R. □

Пример 3.5 показывает, что графическое представление запросов на QUEL может быть расширено для конъюнктивных запросов с больше чем одного RANGE-оператора для некоторой реляционной схемы. Чтобы осуществить это расширение, достаточно внести в таблицу доступа дополнительные строки для этой реляционной схемы столько раз, сколько RANGE-операторов для нее заданы в запросе. Легко видеть, что по такому "расширенному" графическому представлению можно построить таблицу с помощью алгоритма 3.1. В такой таблице будут участвувать столько строк для данной реляционной схемы, сколько RANGE-операторов заданы для нее в запросе.

В главе 4 предложен алгоритм оптимизации в полиномиальном времени только тех таблиц, в который для каждой реляционной схемы участвует наибольше одна строка. Так как наша цель - исследовать проблему оптимизации, здесь мы не будем касаться вышеупомянутого расширения определения 3.5 и алгоритма 3.1.

## 3.4. Виды конъюнктивных запросов на QUEL

С помощью алгоритма 3.1 можно построить класс таблиц. В этой части рассмотрим класс запросов, для которых можно построить таблицы с помощью алгоритма 3.1.

Как видно из синтаксиса языка QUEL [SWKH76], запросы QUEL представляют собой выражения вида:

RANGE OF $a_1$ IS $R_1$

...  (3.4)

RANGE OF $a_n$ IS $R_n$

RETRIEVE (A) WHERE (F)

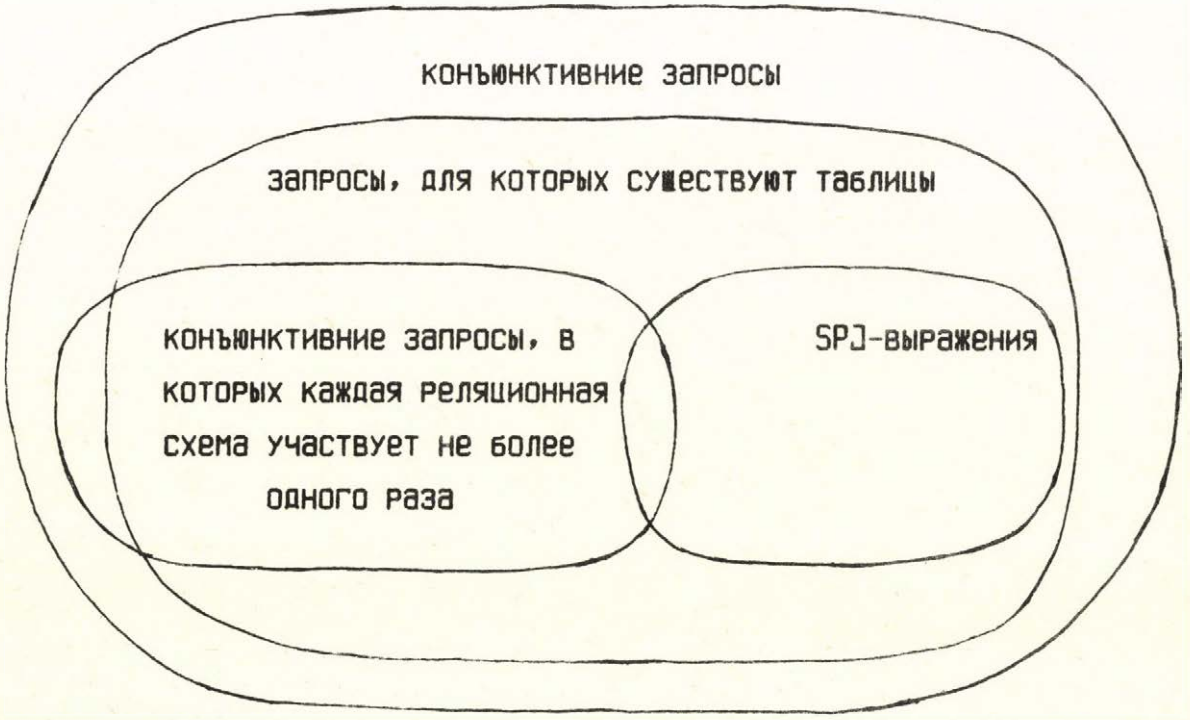
В [Улл80] показано, что множество кортежей, которое представляет собой ответ на этот запрос, задается с помощью выражения:

$$\pi_A(\sigma_F(R_1 \times R_2 \times \ldots \times R_n)) \qquad (3.5)$$

Легко видеть, что запрос (3.4) - соответственно (3.5) является конъюнктивным тогда и только тогда, когда F - формула [Cod71], в которой участвуют только конъюнкции.

Конъюнктивные запросы на языке QUEL, содержащие ровно по одному RANGE-оператору для каждой реляционной схемы, являются запросами вида (3.4), в которых $R_I \neq R_J$ при $I \neq J$ для $1 \leq I \leq n$, $1 \leq J \leq n$. Будет показано, что эти запросы можно отнести к нескольким разным типам.

Рассмотрим снова часть схемы на фиг. 3:



Видно, что конъюнктивные запросы на QUEL, содержащие по одному RANGE-оператору для каждой реляционной схемы, отнесены к трем разным видам запросов:

а) конъюнктивные запросы, для которых нельзя построить таблицы;

б) запросы, для которых можно построить таблицу, но не существует SPJ-выражения;

в) запросы, для которых существуют SPJ-выражения.

Остановимся отдельно на каждом из этих видов запросов.

### 3.4.1. Конъюнктивные запросы, для которых нельзя построить таблицы

Это запросы вида (3.4), для которых объектный список запроса содержит определенный атрибут больше чем один раз.

ПРИМЕР 3.9. Рассмотрим следующий конъюнктивный запрос для реляционных схем $R_1 = \{A,B,C\}$ и $R_2 = \{A,D,E\}$:

$$\{ a_1 a_2 \mid R_1(a_1, b_1, b_2) \,\&\, R_2(a_2, b_3, b_4) \,\&\, (a_1 > a_2) \} \qquad (3.4)$$

По этому конъюнктивному запросу нельзя построить таблицу, потому что свободные переменные $a_1$ и $a_2$ надо поставить в одном и том же столбце в резюме. □

Для запросов этого вида нельзя построить таблицу (и нельзя исполнить пункт 1 алгоритма 3.1, потому что нельзя записать в одном столбце резюме таблицы разные свободные переменные). Следовательно, до сих пор не существуют средства и методы оптимизации запросов этого вида и на них можно отвечать с помощью выражения (3.5).

### 3.4.2. Запросы, для которых можно построить таблицу, но не существует SPJ-выражения

Как показано в [ASU79], таблица

| A | B |
|---|---|
| $a_1$ | $a_2$ |
| $a_1$ | $b_2$ |
| $b_1$ | $a_2$ |
| $b_1$ | $b_2$ |

(3.6)

не может быть получена ни из одного SPJ-выражения.

ПРИМЕР 3.10. Покажем конъюнктивный запрос на QUEL, которому соответствует таблица (3.6):

RETRIEVE $R_1$.A,$R_2$.B (3.7)

WHERE ($R_2$.A=$R_3$.A) AND ($R_1$.B=$R_3$.B)

Так как некоторые определения схемы базы данных требуют чтобы отдельные реляционные схемы представляли собой разные множества атрибутов [Mai83], поставим требование $R_1$ = {A,B,C}, $R_2$ = {A,B,D} и $R_3$ = {A,B,E}. Тогда запросу (3.7) с помощью алгоритма 3.1 будет сопоставлена таблица

| A | B | C | D | E | (3.7′) |
|---|---|---|---|---|---|
| $a_1$ | $a_2$ | | | | |
| $a_1$ | $b_2$ | $b_3$ | | | ($R_1$) |
| $b_1$ | $a_2$ | | $b_4$ | | ($R_2$) |
| $b_1$ | $b_2$ | | | $b_5$ | ($R_3$) |

Докажем, что эту таблицу, так же как и таблицу (3.6), нельзя получить с помощью никакого SPJ-выражения по известному из [ASU79] алгоритму построения таблиц для данных SPJ-выражений.

Допустим, что существует некоторое SPJ-выражение, для которого можно построить таблицу (3.7′). Очевидно это выражение должно содержать две операции соединения ($\bowtie$). Рассмотрим какие строки из (3.7′) могут быть получены на первом шагу построения таблицы.

Строки ($R_1$) и ($R_2$) не могут быть получены из некоторого выражения $R_1 \bowtie R_2$ - потому что все их символы разные. Строки ($R_2$) и ($R_3$) не могут быть получены на первом шагу построения, потому что в таком случае строка ($R_1$) добавлена на втором шагу построения и потом сделана селекция $\sigma_{B=a1}$ - и тогда все символы в столбце B должни быть одинаковыми. Аналогично видно, что строки ($R_1$) и ($R_3$) не могут быть получены на первом шагу построения таблицы (3.7′).

Следовательно, не существует SPJ-выражение, из которого можно получить таблицу (3.7′) с помощью алгоритма 3.1. □

### 3.4.3.Запросы, для которых существуют SPJ-выражения

Покажем, что для некоторых конъюнктивных запросов на QUEL

вида (3.4) существуют SPJ-выражения, с помощью которых можно получить ответ запроса.

Пусть реляционные схемы $R_1, R_2, \ldots, R_n$ выражения (3.4) - соединяемые (Joinable) относительно множеств атрибутов $X_1, X_2, \ldots, X_{n-1}$, где $X_1 = R_1 \cap R_2$, $X_2 = R_2 \cap R_3$, ..., $X_{n-1} = R_{n-1} \cap R_n$.

Если $X_i = \{A_{i1}, A_{i2}, \ldots, A_{ir}\}$, через $R_i.X_i = R_{i+1}.X_i$ будем обозначать все условия

$R_i.A_{i1} = R_{i+1}.A_{i1}$, $R_i.A_{i2} = R_{i+1}.A_{i2}$, ..., $R_i.A_{ir} = R_{i+1}.A_{ir}$.

Легко видеть, что при помощи формул

$$\pi_A (\sigma_F (R_1 \times R_2 \times \ldots \times R_n)) \qquad (3.8)$$

и

$$\pi_A (\sigma_F (R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n)) \qquad (3.9)$$

получаются одни и те же кортежи для всех состояний универсума U тогда и только тогда, когда

$$F = F' \& (R_1.X_1 - R_2.X_2) \& (R_2.X_2 - R_3.X_2) \& \ldots \& (R_{n-1}.X_{n-1} - R_n.X_{n-1})$$

Следовательно, для некоторых конъюнктивных запросов на QUEL с по одному RANGE-оператору для данных реляционных схем существуют SPJ-выражения, описывающие ответ на запрос.

ТЕОРЕМА 3.1. Пусть q - конъюнктивный запрос на QUEL, который содержит наибольше по одному RANGE-оператору для данных реляционных схем. Пусть существует SPJ-выражение, с помощью которого можно получить ответ на q. Пусть для этого SPJ-выражения строится таблица T с помощью известного алгоритма из [ASU79]. Тогда при помощи алгоритма 3.1 строится таблица T', идентичная таблице T с точностью до переименования символов.

Доказательство проводится индуктивным образом по числу операции $\pi$, $\sigma$ и $\bowtie$, которые содержатся в выражении (3.9).

Пусть выражение (3.9) не содержит операций $\pi$, $\sigma$ и $\bowtie$ - т.е. выражение (3.9) содержит только имя некоторой реляционной схемы. Тогда видно, что алгоритм 3.1 строит для выражения (3.9) таблицу, идентичная (с точности до переименования символов) таблице построенной для (3.9) с помощью алгоритма 1.1.

Допустим, что для выражения вида (3.9) с n операций $\pi$, $\sigma$ и $\bowtie$ алгоритмы 1.1 и 3.1 строят идентичные таблицы. Легко получить, что для выражения вида (3.9) с n+1 операций $\pi$, $\sigma$ и $\bowtie$ алгоритмы 1.1 и 3.1 строят идентичные таблицы. Для доказательства этого факта используются сформулированные выше условия эквивалентности выражения вида (3.8) и (3.9). □

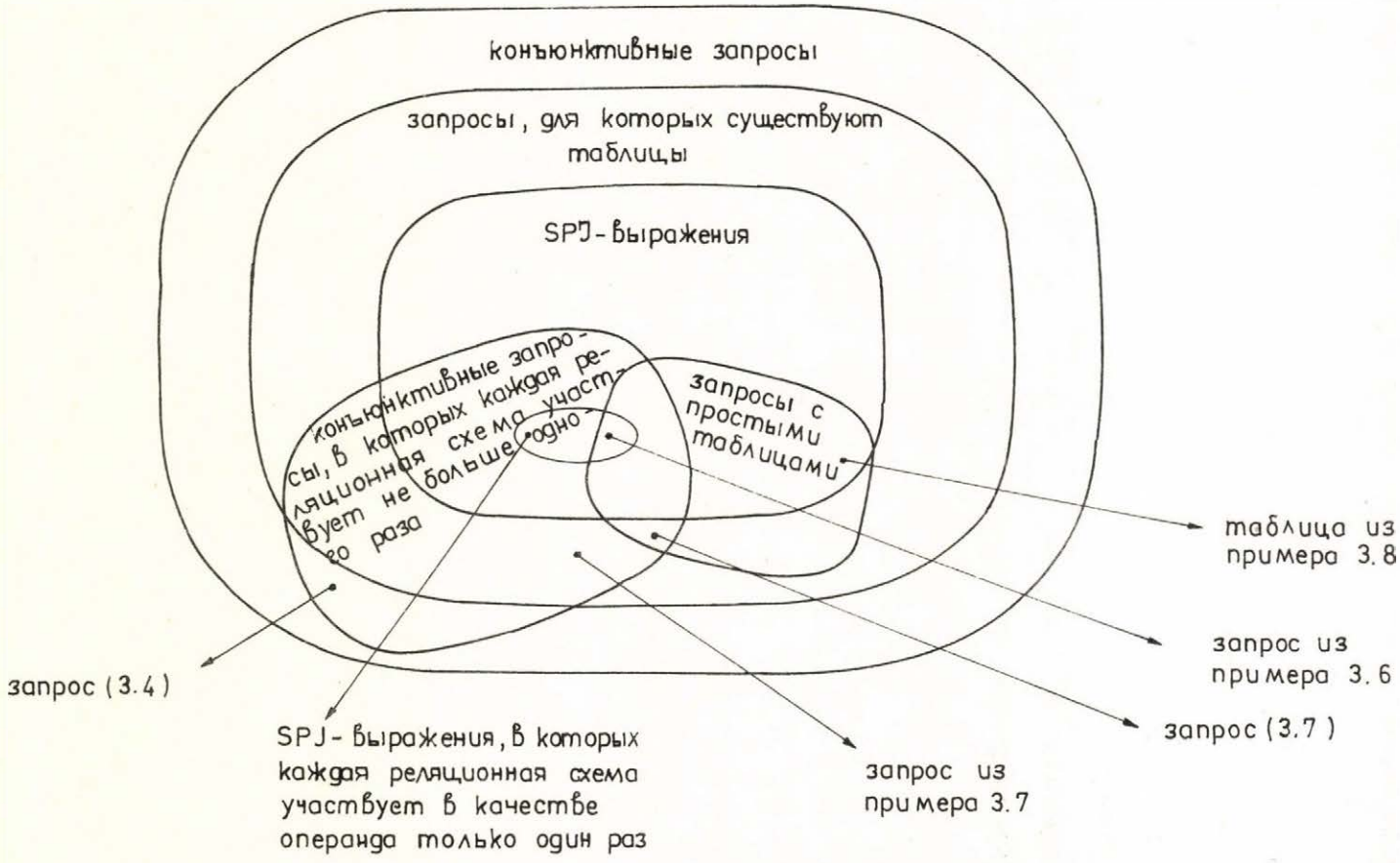
С помощью алгоритма 3.1 мы можем построить класс таблиц. Для некоторых из них существуют SPJ-выражения. Как уже известно, SPJ-выражения можно представлять в виде формул (согласно определению 1.1), в которых имена реляционных схем участвуют как операнды. Видно, что класс таблиц, которые можно построить по алгоритму 3.1, содержит таблицы, которые можно построить по алгоритму из [ASU79] для SPJ-выражения, где каждая реляционная схема участвует как операнд не более одного раза.

Пример 3.6 показывает, что класс конъюнктивных запросов на QUEL с по одному RANGE-оператору пересекается с классом запросов, для которых существует простая таблица; примеры 3.7 и 3.8 показывают, что ни один из обоих классов не содержит другого в качестве собственного подмножества.

Следовательно, схему из фиг.3 можно уточнить, как это показано на фиг. 3.1.

Теорема 3.1 позволяет нам ввести следующее определение:

ОПРЕДЕЛЕНИЕ 3.6. Пусть дан конъюнктивный запрос q на языке QUEL с одним RANGE-оператором для каждой реляционной схемы.




фиг. 3.1. Классы запросов на процедурных и декларативных реляционных языках

Пусть объектный список этого запроса не содержит повторяющихся атрибутов. Пусть запросу $q$ сопоставлено графическое представление $q_G$. Тогда под результатом (ответом) на запрос $q$ будем подразумевать отношение, заданно с помощью построенной по алгоритму 3.1 таблицы графического представления $q_G$. □

Определение 3.6 - коректно, потому что (как показывает теорема 3.1), в случае совпадения выражений вида (3.8) и (3.9) алгоритмы 1.1 и 3.1 строят идентичные с точностью до переименования символов таблицы.

# Г Л А В А  Ч Е Т В Е Р Т А Я

# ОПТИМИЗАЦИЯ КОНЪЮНКТИВНЫХ ЗАПРОСОВ НА QUEL

В этой главе рассматривается проблема оптимизации таблиц, построенных по алгоритму 3.1.

В параграфе 4.1 показано, что построенные по алгоритму 3.1 таблицы являются минимальными таблицами при сильной эквивалентности.

В параграфе 4.2 введено представление этих таблиц в виде гиперграфов. Показано, что таблицы с ациклическими гиперграфами можно минимизировать в полиномиальном периоде времени с помощью модификации алгоритма редукции гиперграфов.

В параграфе 4.3 показано, что таблицы с циклическими гиперграфами тоже можно минимизировать в полиномиальном периоде времени.

После введения представления запросов в виде гиперграфов становится возможным изучать разные виды запросов - A и B-ациклические запросы и циклические запросы. В параграфе 4.4 рассмотрены виды запросов, которые можно сформулировать в разных реляционных схемах. Введено понятие D-ациклического гиперграфа схемы реляционной базы данных. Показано, что в случае D-ациклических схем, можно сформулировать только D-ациклические запросы.

## 4.1. Сильная эквивалентность и оптимизация конъюнктивных запросов

Как сказано в определении 1.11, минимальной таблицей для данной таблицы T называется таблица, содержащая минимальное число строк и эквивалентная таблице T. Процесс нахождения минимальной таблицы для данной таблицы T называется оптимизацией таблицы T.

В [ASU79] и [Ull82] показано, что таблицы $T_1$ и $T_2$ являются сильно эквивалентнимы тогда и только тогда, когда существуют содержащие отображения (см. определения 1.10) $h_1$ и $h_2$ такие, что $h_1:T_1 \rightarrow T_2$, $h_2:T_2 \rightarrow T_1$ и $h_1, h_2$ сохраняют маркеры всех строк. Сразу можно доказать следующую теорему:

ТЕОРЕМА 4.1. Пусть T - таблица, построенная по алгоритму 3.1. Тогда T является минимальной таблицей при сильной эквивалентности.

ДОКАЗАТЕЛЬСТВО. Таблица T содержит только одну строку для каждого данного маркера. Поэтому из T нельзя убрать некоторую строку с маркером $(R_1)$, потому что в полученной таблице уже не будет участвовать строка с маркером $(R_1)$.

Следовательно, T является минимальной таблицей при сильной эквивалентности. □

## 4.2. Слабая эквивалентность и оптимизация конъюнктивных ациклических запросов

Как сказано в параграфе 1.5, таблицы $T_1$ и $T_2$ являются слабо эквивалентными тогда и только тогда, когда существуют содержащие отображения $h_1: T_1 \to T_2$ и $h_2: T_2 \to T_1$, причем $h_1$ и $h_2$ не должны учитывать маркеры разных строк таблиц (так как $\cup$ является маркером всех строк).

Нужно отметить, что пустые символы в данной таблице при слабой эквивалентности всегда являются разными связанными переменными, которые встречаются только один раз в этой таблице. Поэтому, если нужно, мы будем заполнять эти пустые места связанными переменными $d_1, d_2, \ldots$ . В этом случае связанные переменные $d_1, d_2, \ldots$ поставляются в строке таблицы в тех столбцах, которые не соответствуют атрибутам из реляционной схемы, с именем которой можно маркировать строку. Но как часто делается в [ASU79] и в [Ull82], иногда мы не будем заполнять эти элементы таблиц (будем оставлять их пустыми).

В этом параграфе (и в параграфах 4.3 и 4.4) будем рассматривать запросы вида (1.3), для которых "$\theta$" везде является "$=$". С помощью алгоритма 3.1 этим запросам сопоставляются таблицы с пустыми списками ограничений.

Опишем алгоритм минимизации запросов при слабой эквивалентности, который напоминает алгоритм редукции гиперграфов. Введем понятие изолированного символа, подобно понятию изолированной вершиной гиперграфа (см. определение 2.4).

ОПРЕДЕЛЕНИЕ 4.1. Пусть b (d) - связанная переменная, которая участвует только в одной строке данной таблицы T. Тогда b (d) будем называть изолированным символом для таблицы T. □

Пусть $A=\{a_1,a_2,\ldots\}$ - множество всех свободных переменных, $B=\{b_1,b_2,\ldots\}$ - множество всех связанных переменных, $C=\{c_1,c_2,\ldots\}$ - множество всех констант, $D=\{d_1,d_2,\ldots\}$ - множество всех связанных переменных, которые служат для заполнения строк таблиц. (Отметим, что в силу определения 4.1 все элементы множества D - изолированные символы для таблиц, в которых они появляются). Рассмотрим следующий алгоритм:

АЛГОРИТМ 4.1. Редукция строк данной таблице T.

ВХОД: Таблица T, построенная по алгоритму 3.1 для конъюнктивного запроса на QUEL с одним RANGE-оператором для каждой реляционной схемы. (Отметим, что в этих таблицах не вносятся связанные переменные $d,d_1,d_2,\ldots$ и так далее. Их места заполнены пустыми символами).

ОПЕРАЦИЯ 1. Зачеркнуть все изолированные символы, которые появляются только один раз в строках таблицы T.

ОПЕРАЦИЯ 2. Если какая-то строка $r_1$ таблицы T целиком содержит другую строку $r_2$ таблицы T, зачеркнуть строку $r_2$ таблицы T.

МЕТОД: Применять операции 1 и 2 в произвольном порядке, сколько раз возможно.

ВЫХОД: Таблица T', которая содержит меньше строк, чем таблица T. У таблицы T' такое же резюме как у таблицы T. □

Отметим, что для алгоритма 4.1 изолированными символами

могут быть только элементы множеств B и D. В силу этого вышеописанный алгоритм представляет собой модификацию алгоритма редукции гиперграфов - где все вершины можно зачеркнуть с помощью операции 1, если вершины встречаются только один раз в гиперграфе.

Как и алгоритм 2.1, алгоритм 4.1 всегда заканчивает работу в полиномиальном периоде времени. Дальше, аналогично определению 2.7 редукции гиперграфов, мы будем говорить о редукции таблиц.

Введем следующие обозначения: если таблица T′ получается от данной таблицы T после однократного применения операции 1 алгоритма 4.1, будем обозначать это через $T \rightarrow_N T'$. Если таблица T′ получается от данной таблицы T после однократного применения операции 2 алгоритма 4.1, будем обозначать это через $T \rightarrow_E T'$. Запись $T \rightarrow_* T'$ означает что таблица T′ получается от таблицы T путем многократного применения операций 1 и 2 алгоритма 4.1.

ЛЕММА 4.1. Если T - таблица и T′ - таблица такая, что $T \rightarrow_* T'$, то тогда $T \text{ --}_{ш}\ T'$.

ДОКАЗАТЕЛЬСТВО. Для T и T′ можно построить цепочку таблиц

$$T = T_1, T_2, \ldots, T_{k-1}, T_k = T'$$

где каждая таблица $T_i$, $2 \le i \le k$, получена из предшествующей таблицы $T_{i-1}$ путем одного применения операцию 1 или операцию 2 алгоритма 4.1. Рассмотрим два случая:

а) $T_{i-1} \rightarrow_N T_i$. Тогда $T_{i-1}$ и $T_i$ имеют одинаковое число строк и различаются только по изолированным символам в разных строках. Следовательно, существуют содержащее отображение $h_1$ из символов $T_{i-1}$ в символы $T_i$ и содержащее отображение $h_2$ из

символов $T_i$ в символы $T_{i-1}$ так, что:

- $h_1$ и $h_2$ сохраняют свободные переменные и константы из $T_{i-1}$ и $T_i$;

- $h_1$ и $h_2$ сохраняют связанные переменные, которые не являются изолированными символами в $T_{i-1}$ и $T_i$;

- $h_1$ отображает изолированные связанные переменные из **B** таблицы $T_{i-1}$ соответственно в изолированные связанные переменные из **D** таблицы $T_i$, а $h_2$ отображает изолированные связанные переменные из **D** таблицы $T_i$ соответственно в изолированные связанные переменные из **B** таблицы $T_{i-1}$;

- $h_1$ и $h_2$ сохраняют связанные переменные из **D**, которые участвуют одновременно как в $T_{i-1}$, так и в $T_i$.

Следовательно, $T_{i-1} \dashrightarrow_{\text{Ш}} T_i$.

б) $T_{i-1} \rightarrow_E T_i$. Тогда $T_i$ содержит на строку меньше, чем $T_{i-1}$. Пусть строки $T_{i-1}$ - $r_1, r_2, \ldots, r_s, r_{s+1}$, а строки $T_i$ - $r_1, r_2, \ldots, r_s$. (Здесь через $r_{s+1}$ обозначили ту строку таблицы $T_{i-1}$, символы которой целиком содержатся в некоторой строке с маркером $r_p$). Тогда можно построить следующие содержащие отображения между символами таблицы $T_{i-1}$ и $T_i$:

- $h_1$ из символов таблицы $T_{i-1}$ в символы таблицы $T_i$; $h_1$ отображает строки $r_1, r_2, \ldots, r_s$ в их самих, а строка $r_{s+1}$ отображается в строку $r_p$ таблицы $T_i$;

- $h_2$ из символов таблицы $T_i$ в символы таблицы $T_{i-1}$; $h_2$

отображает каждый символ (каждую строку) в себя.

Отображения $h_1$ и $h_2$ показывают, что $T_{i-1} \text{ --}_w\ T_i$.

Следовательно, для цепочки таблиц $T\text{-}T_1, T_2, \ldots, T_{k-1}, T_k\text{-}T'$ получается, что $T_{i-1} \text{ --}_w\ T_i$ для $2 \leq i \leq k$. Тогда $T \text{ --}_w\ T'$. □

ОПРЕДЕЛЕНИЕ 4.2. Пусть T - таблица, над которой работает алгоритм 4.1. Пусть на выходе алгоритма получается таблица T'. Тогда таблицу T' будем называть редуцированной таблицей для таблицы T. □

Если T - таблица, будем обозначать ее редуцированную таблицу через GR(T).

Докажем, что для данной таблицы T, таблица GR(T) не зависит от порядка применения операции 1 и операции 2 алгоритма 4.1.

Нахождение GR(T) является примером системы замещения (replacement system) [Mal83].

ОПРЕДЕЛЕНИЕ 4.3. [Mal83] Системой замещения называется упорядоченная пара (Q,->), где Q - множество объектов, а -> - нерефлексивная, двоичная реляция над Q. □

Будем называть -> трансформацией. В качестве примера системы замещения можно рассмотреть множество таблиц, построенных по алгоритму 3.1 и трансформации $\rightarrow_N$ или $\rightarrow_E$, определенные соответственно с помощью операций 1 и 2 алгоритма 4.1.

ОПРЕДЕЛЕНИЕ 4.4. [Mal83] Через $\overset{*}{\rightarrow}$ будем обозначать рефлексивное, транзитивное замыкание реляции ->. □

В качестве примера можно рассмотреть трансформацию $\rightarrow_*$, которая означает многократное применение в произвольном порядке трансформаций $\rightarrow_N$ и $\rightarrow_E$.

ОПРЕДЕЛЕНИЕ 4.5. [Mal83] Для данной системы замещения (Q,->) объект $p \in Q$ называется <u>нередуцируемым</u>, если из $p \rightarrow^* q$ следует, что $p = q$. □

Пусть через $q_0$, $q_0 \in Q$ обозначим некоторый нередуцируемый объект множества Q.

ОПРЕДЕЛЕНИЕ 4.6. [Mal83] Система замещения (Q,->) является <u>конечной</u> (finite), если для каждого объекта $p \in Q$ существует константа c, зависящая от p так, что если $p \overset{*}{\rightarrow} q_0$ для l шагов, тогда $l \leq c$. □

В качестве примера конечной системы замещения можно рассмотреть множество таблиц, построенных по алгоритму 3.1, и некоторую из трансформаций $\rightarrow_N$ и $\rightarrow_E$ алгоритма 4.1. Так как при трансформациях $\rightarrow_N$ и $\rightarrow_E$ только зачеркиваются символы таблиц, то от каждой таблицы T можно дойти до нередуцируемой таблицы GR(T), при чем число шагов не больше чем число символов в T.

ОПРЕДЕЛЕНИЕ 4.7. [Mal83] Конечная система замещения (Q,->) является <u>конечной системой Черча-Россера</u> (finite Church-Rosser system), если для каждого объекта $p \in Q$ имеем: если $p \overset{*}{\rightarrow} q_1$ и $p \overset{*}{\rightarrow} q_2$, где $q_1$ и $q_2$ - нередуцируемые, то $q_1 = q_2$. □

В качестве примера нередуцируемого объекта можно рассмотреть пустую запись, которая является нередуцируемым объектом в множестве ациклических гиперграфов при применении операций 1 и 2 алгоритма 2.1. Приведенная в главе 2 теорема 2.1 показывает, что независимо от порядка применения операций 1 и 2 в алгоритме 2.1, ациклические гиперграфы всегда редуцируются до пустой записи.

ТЕОРЕМА 4.2. [Mai83] Система замещения $(Q,\to)$ является конечной системой Черча-Россера тогда и только тогда, когда

1. $(Q,\to)$ - конечная система замещения;
2. Для каждого объекта $p\in Q$, если $p\to q_1$ и $p\to q_2$, то существует $q\in Q$ такое, что $q_1 \overset{*}{\to} q$ и $q_2 \overset{*}{\to} q$. □

ЛЕММА 4.2. GR(T) не зависит от порядка применения операций 1 и 2 алгоритма 4.1.

ДОКАЗАТЕЛЬСТВО. Пусть $Q_0$ - множество всех таблиц, которые могут быть построены по алгоритму 3.1. Пусть $\to_0$ - трансформация $\to_N$ или $\to_E$. Покажем, что $(Q_0,\to_0)$ является конечной системой Черча-Россера.

1. Покажем, что $(Q_0,\to_0)$ является конечной системой. Трансформации $\to_N$ и $\to_E$ всегда зачеркивают символы в данных таблицах. Следовательно, от данной таблицы T всегда можно дойти до GR(T) для l шагов, $l\le c$, при чем константа c - число символов в данной таблице T.

Для конечной системы $(Q_0,\to_0)$ можно найти и константу $c_0$, независящую от данного объекта $T\in Q_0$. Пусть $R=\{R_1,R_2,\ldots,R_m\}$ -

схема базы данных, для которой рассматриваются запросы с таблицами $Q_0$. Пусть $U$ состоит из n атрибутов. Тогда для $Q_0$ существует константа $c_0 = n.m$ (Здесь $c_0$ - максимальное число символов в таблицах из $Q_0$).

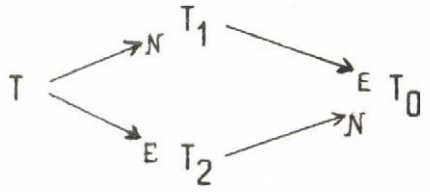
2. Пусть $T \in Q_0$. Допустим, что возможно $T \to_0 T_1$ и $T \to_0 T_2$, $T_1 \neq T_2$. Докажем, что тогда существует $T_0 \in Q_0$ так, что $T_1 \overset{*}{\to}_0 T_0$ и $T_2 \overset{*}{\to}_0 T_0$.

Исследуем все случаи:

а) Пусть $T \to_N T_1$ и $T \to_E T_2$. Пусть трансформация $\to_E$ применяется для пары строк $r_1$ и $r_2$ из T и пусть символы $r_2$ целиком содержатся в символах $r_1$. Тогда трансформация $\to_N$ не может применяться над символами из $r_2$ в T (которые не изолированы, потому что участвуют и в строке $r_1$). Следовательно, и в таблице $T_1$ присутствуют строки $r_1$ и $r_2$ и можно применять трансформацию $\to_E$ над ними.

Пусть $T_0 : T_1 \to_E T_0$. Тогда видно, что $T_2 \to_N T_0$.

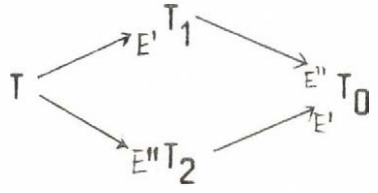
Следовательно, для T, $T_1$, $T_2$ и $T_0$ исполнено:




б) $T \to_N T_1$ и $T \to_N T_2$. Этот случай невозможен, потому что с помощью операции 1 алгоритма 4.1 зачеркиваются одновременно все символы, которые являются изолированными для T (на данном уровне редукции таблицы T).

в) Пусть $T \rightarrow_E' T_1$ и $T \rightarrow_E'' T_2$. Легко видеть, что трансформации $\rightarrow_E$ - комутативные. Пусть $T_0$: $T_1 \rightarrow_E'' T_0$; здесь с помощью трансформации $\rightarrow_E''$ в $T_1$ поглощаются те строки, которые поглощаются в процессе $T \rightarrow_E' T_2$. Тогда $T_2 \rightarrow_E' T_0$.

Следовательно, для T, $T_1$, $T_2$ и $T_0$ исполнено:




Следовательно, для каждой таблицы $T \in Q_0$ существует точно один нередуцируемый объект GR(T) из $T_0$. T трансформируется в GR(T) путем многократного применения в произвольном порядке операций 1 и 2 алгоритма 4.1. □

СЛЕДСТВИЕ 4.1. Из лем 4.1 и 4.2 следует, что $T \dashv\!\!\rightarrow_{\Sigma} GR(T)$.

□

Для дальнейшего рассмотрения проблемы оптимизации необходимо ввести представление запросов в виде гиперграфов.

Введем понятия повторяющегося символа.

ОПРЕДЕЛЕНИЕ 4.8. [ASU79] Повторяющимся символом в данном столбце таблицы T называется связанная переменная, появляющаяся больше чем в одной строке данного столбца таблицы T. □

Отметим, что полученные по алгоритму 3.1 таблицы содержат для каждой реляционной схемы не больше чем одну строку, маркированную именем этой схемы; следовательно, повторяющимися

символами могут быть только те связанные переменные, которые участвуют в столбцах таблицы, соответствующих атрибутам, входящим больше чем в одну реляционную схему.

Пусть $A=\{a_1,a_2,\ldots\}$ - множество свободных переменных, $C=\{c_1,c_2,\ldots\}$ - множество констант, $B=\{b_1,b_2,\ldots\}$ - множество всех связанных переменных, которые могут быть повторяющимися символами в некоторой таблице, и $D=\{d_1,d_2,\ldots\}$ - множество всех связанных переменных, которые служат для заполнения строк таблиц и поэтому всегда являются изолированными символами. Пусть для каждого элемента множеств $A,B,C$ и $D$ указано, в домене какого атрибута $U$ этот элемент можно менять: например, $a_1 \in dom(A_3)$, $a_2 \in dom(A_5)$, $b_1 \in dom(A_2)$ и так далее.

ОПРЕДЕЛЕНИЕ 4.9. Для данной таблицы T, построенной по алгоритму 3.1, определим гиперграф H(T), соответствующий таблице T.

Пусть $A(T)$, $C(T)$ и $B(T)$ соответственно свободные переменные, константы и связанные переменные, которые могут быть повторяющимися символами в таблице T. Пусть $H(T)=(V,E)$. Тогда множество вершин $V$ задается следующим образом:

$$V = A(T) \cup B(T) \cup C(T) .$$

Каждой строке таблицы T сопоставляется ребро гиперграфа H(T):

$$E = \{ e \mid e=\{s_1,s_2,\ldots,s_p\}, s_i \in A(T) \cup B(T) \cup C(T) \text{ для } 1 \le i \le p \text{ и } \{s_1,s_2,\ldots,s_p\} \text{ является строкой из T.} \} . \quad \square$$

ПРИМЕР 4.1. Пусть $U=\{A,B,C,D\}$ и $R=\{AB,BC,CD,AD\}$. Рассмотрим запрос

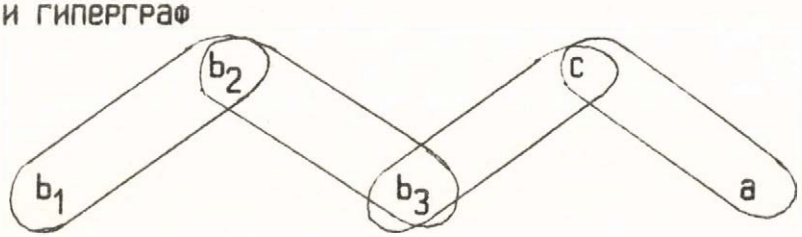
```
RETRIEVE (AD.A)
   WHERE (CD.D-'c')
     AND (AB.B-BC.B) AND (BC.C-CD.C) AND (CD.D-AD.D) .
```

Этому запросу соответствует таблица

| A | B | C | D | | | A | B | C | D | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| a | | | | | | a | | | | | |
| $b_1$ | $b_2$ | | | (AB) | | $b_1$ | $b_2$ | $d_1$ | $d_2$ | (AB) | |
| | $b_2$ | $b_3$ | | (BC) | | $d_3$ | $b_2$ | $b_3$ | $d_4$ | (BC) | |
| | | $b_3$ | c | (CD) | | $d_5$ | $d_6$ | $b_3$ | c | (CD) | |
| a | | | c | (AD) | | a | $d_7$ | $d_8$ | c | (AD) | |

(4.1)

и гиперграф




(4.2)

В вышеуказанной таблице связанные переменные $d_1$-$d_8$ заполняют в каждой строке столбцы универсума, которые не участвуют в соответствующей реляционной схеме. □

Как известно, в [MaUl84] исследованы таблицы, которые могут быть сопоставлены ациклическим гиперграфам. В этом сопоставлении каждому ациклическому гиперграфу соответствует таблица, в столбцах которой встречается только один символ для каждого столбца - или одна свободная переменная, или одна связанная переменная, или одна константа. Такая таблица является всегда простой. В [MaUl84] показано, что для таких таблиц минимизацию можно осуществлять путем модифицированной

редукции данной таблицы.

Наша цель однако - исследовать проблему оптимизации запросов, заданных при помощи декларативного языка QUEL. Таким образом мы ставим себе целью изучить процесс оптимизации данной таблицы, которая (как показывает пример 3.7) может и не быть простой таблицей и поэтому может не принадлежать множеству таблиц, соответствующих ациклическим гиперграфам. Следовательно, сопоставляя гиперграф данной таблице, мы ставим себе задачу, обратную той, которая поставлена в [MaUl84]. Нам предстоит доказать, что и таблицы исследованных нами запросов на QUEL можно минимизировать с помощью редукции по алгоритму 4.1, несмотря на то, что в их столбцах можно располагать разные символы. Исследованные нами таблицы отличаются от таблиц ациклических гиперграфов в [MaUl84] и от простых таблиц в [ASU79]. На следующем примере покажем почему нельзя использовать технику доказательств из [MaUl84].

ПРИМЕР 4.2. Следуя [MaUl84], гиперграфу (4.2) надо сопоставить таблицу

| $B_1$ | $B_2$ | $B_3$ | $C_1$ | $A$ |
|---|---|---|---|---|
| | | | | $a$ |
| $b_1$ | $b_2$ | | | |
| | $b_2$ | $b_3$ | | |
| | | $b_3$ | $c$ | |
| | | | $c$ | $a$ |

(4.3)

Известно, что гиперграф (4.2) соответствует таблице (4.1) в силу определения 4.9, а в некотором смысле таблица (4.3) представляет собой "развертку" таблицы (4.1), так как столбцы для элементов a и $b_1$ отделены друг от друга. По этой причине

таблицы (4.1) и (4.3) имеют разное число столбцов, а мы не можем проверять эквивалентность таблиц с разным числом столбцов. □

Поэтому в процессе исследования введеные нами таблицы, полученные по алгоритму 3.1, и соответствующие им гиперграфы, нельзя применить использованных в [ASU79] и [MaUl84] методов доказательств.

Легко видеть, что если данный запрос на QUEL является связанным (см. определение 1.4), то и гиперграф, сопоставленный таблице этого запроса в силу определения 4.9, тоже является связанным.

Ниже воспользуемся следующими обозначениями: пусть T - таблица, построенная по алгоритму 3.1. Через H(T) будем обозначать гиперграф, сопоставленный таблице T в силу определения 4.9. Если r - строка таблицы T, через e(r) будем обозначать ребро гиперграфа H(T), соответствующее строке r.

ОПРЕДЕЛЕНИЕ 4.10. Данную таблицу T будем называть ациклической, если H(T) является ациклическим гиперграфом. □

ОПРЕДЕЛЕНИЕ 4.11. Данную таблицу T будем называть циклической, если H(T) является циклическим гиперграфом. □

ОПРЕДЕЛЕНИЕ 4.12. Пусть T - таблица со строками $r_1, r_2, \ldots, r_s$. Циклической компонентой таблицы T будем называть множество строк $\{r_{i1}, r_{i2}, \ldots, r_{ik}\}$ такое, что $H(\{r_{i1}, r_{i2}, \ldots, r_{ik}\})$ является циклическим гиперграфом. □

ОПРЕДЕЛЕНИЕ 4.13. Пусть $T_0$ -нередуцируемая таблица и $H(T_0)$ -

связанный гиперграф. Строку $r \in T_0$ будем называть крайней, если после удаления ребра $e(r)$ из $H(T_0)$, полученный гиперграф продолжает оставаться связанным. □

В качестве примера крайней строки можно рассмотреть таблицу (4.1) и ее гиперграф (4.2). В таблице (4.1) строка $\langle b_1, b_2 \rangle$ является крайней.

ОПРЕДЕЛЕНИЕ 4.14. Если некоторая свободная переменная a участвует только один раз в строках данной таблицы T, эту свободную переменную будем называть изолированной свободной переменной для T. □

ОПРЕДЕЛЕНИЕ 4.15. Если некоторая константа c участвует только один раз в строках данной таблицы T, эту константу будем называть изолированной константой для T. □

В качестве примера рассмотрим таблицу (4.1). В таблице (4.1) свободная переменная a является изолированной свободной переменной. Константа $c_1$ в таблице (4.1) не является изолированной константой - она встречается в двух строках таблицы (4.1).

ЛЕММА 4.3. Пусть T - таблица; пусть $\{s_1, s_2, \ldots, s_p\}$ - все символы крайних строк таблицы GR(T), которые появляются только один раз в таблице GR(T). Тогда $\{s_1, s_2, \ldots, s_p\}$ являются изолированными свободными переменными или изолированными константами для таблицы GR(T).

ДОКАЗАТЕЛЬСТВО. Допустим, что для некоторого символа $s_m$ из

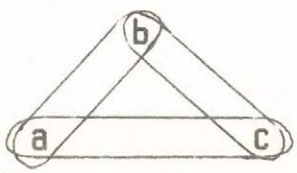
$\{s_1, s_2, \ldots, s_p\}$ исполнено $s_m \in B(GR(T))$. Тогда $s_m$ является изолированным символом для GR(T) и можно еще раз применить операцию 1 алгоритма 4.1. Тогда GR(T) не является нередуцируемой таблицей, что противоречит определению GR(T). Следовательно, если крайние строки GR(T) содержат однократно встречающиеся в таблице GR(T) символы, эти символы являются изолированными свободными переменными или изолированными константами для таблицы GR(T). □

Покажем, что для циклических таблиц все строки могут быть крайними.

ПРИМЕР 4.3. Рассмотрим нередуцируемую таблицу

| a | | |
|---|---|---|
| a | b | |
| | b | c |
| a | | c |

Этой таблице можно сопоставить гиперграф




Видно, что при удалении каждого ребра этого связанного гиперграфа, полученный гиперграф продолжает быть связанным.

□

ЛЕММА 4.4. Если T - ациклическая таблица, то GR(T) - тоже ациклическая таблица.

ДОКАЗАТЕЛЬСТВО. Как показано в лемме 2.2, в процессе редукции ациклического гиперграфа нельзя получить на некотором уровне редукции запись, соответствующую циклическому гиперграфу

(иначе входной ациклический гиперграф нельзя редуцировать до пустой записи).

Операция 1 алгоритма 4.1 слабее, чем операция 1 алгоритма 2.1 - потому, что в алгоритме 4.1 нельзя зачеркнуть все символы, которые появляются только один раз в строках входной таблицы. Поэтому если T - ациклическая таблица, то на всех уровнях редукции с помощью алгоритма 4.1 в силу леммы 2.2 получается также ациклическая таблица.

Следовательно, GR(T) является ациклической таблицей. □

Здесь нужно отметить, что GR(T) может содержать циклическую компоненту (см. определения 2.11). Лемма 4.4 показывает, что если GR(T) содержит циклическую компоненту, то тогда GR(T) содержит и ее поглощающие строки (см. определения 2.6).

ЛЕММА 4.5. Пусть T - таблица и H(T) - связанный гиперграф. Тогда на всех уровнях редукции таблицы T с помощью алгоритма 4.1 получается таблица, которой можно сопоставить связанный гиперграф.

ДОКАЗАТЕЛЬСТВО. См. доказательство леммы 2.1. □

ЛЕММА 4.6. Пусть T - ациклическая таблица и H(T) - связанный гиперграф. Тогда GR(T) содержит только такие крайние строки, в которых участвуют изолированные свободные переменные или изолированные константы.

ДОКАЗАТЕЛЬСТВО. Из леммы 4.4 следует, что GR(T) является ациклической таблицей. Из леммы 4.5 следует, что таблице GR(T) сопоставляется связанный гиперграф H(GR(T)).

Рассмотрим два случая:

а) GR(T) не содержит циклическую компоненту - другими словами, гиперграф H(GR(T)) является B-ациклическим.

Докажем, что в крайних строках таблицы GR(T) участвуют однократно встречающиеся для GR(T) символы.

Допустим, что существует r - крайняя строка для GR(T) такая, что r не содержит однократно встречающихся для GR(T) символы. Тогда e(r)=con(e(r)), т.е. ребро e(r) состоит только из связанных вершин. Ребро e(r) не содержится целиком в другом ребре гиперграфа H(GR(T)) (потому что над гиперграфом H(GR(T)) нельзя применить операцию 2 алгоритма 2.1). Следовательно, в e(r) существуют две вершины $n_1$ и $n_2$ такие, что $n_1 \in e_1 \cap e(r)$ и $n_2 \in e_2 \cap e(r)$, где $e_1$ и $e_2$ - ребра гиперграфа H(GR(T)). Тогда очевидно ребра $e_1$ и $e_2$ связаны с помощью ребра e(r). Но как сказано выше, r является крайней строкой и после ее удаления полученный из H(GR(T)) гиперграф H' продолжает быть связанным. (H' содержит все ребра из H(GR(T) кроме ребра e(r) ). Тогда ребра $e_1$ и $e_2$ связаны в H' с помощью некоторой цепочки ребер

$$e_1=e_1', e_2', \ldots, e_{k-1}', e_k'=e_2,$$

где $e_i' \neq e(r)$ и $e_i' \in H'$ для $1 \leq i \leq k-1$.

Тогда очевидно ребра

$$\{e_1, e_2', \ldots, e_{k-1}', e_2, e, e_1\}$$

образуют циклическую компоненту в H(GR(T)) - как следует из теоремы 2.4, что является противоречием.

Следовательно, каждая крайняя строка r в GR(T) должна содержать некоторые однократно встречающиеся для GR(T) символы. Как показывает лемма 4.3, эти символы могут быть только изолированными свободными переменными или изолированными константами таблицы GR(T).

б) GR(T) содержит циклическую компоненту - другими словами,

гиперграф H(GR(T)) является A-ациклическим. Знаем, что GR(T) содержит и поглощающие строки, соответствующие этой циклической компоненте. Пусть r - крайняя строка из циклической компоненты. Допустим, что соответствующее ребро e(r) гиперграфа H(GR(T)) состоит только из связанных вершин, т.е. e(r)=con(e(r)). Тогда строка r∈GR(T) целиком содержится в некоторой из поглощающих строк циклической компоненты в GR(T) и может быть зачеркнута из GR(T) с помощью применения операции 2 алгоритма 4.1. Следовательно, GR(T) не является нередуцируемой таблицей - что противоречит условию леммы.

Следовательно, каждая крайняя строка из циклической компоненты должна содержать однократно встречающиеся для GR(T) символы. Лемма 4.3 показывает, что они могут быть изолированными свободными переменными или изолированными константами для GR(T). □

ТЕОРЕМА 4.3. Пусть T - ациклическая таблица. Тогда GR(T) - таблица с минимальным числом строк, слабо эквивалентная таблице T.

ДОКАЗАТЕЛЬСТВО. Рассмотрим два случая:

а) Пусть H(T) - связанный гиперграф. Тогда из леммы 4.5 следует, что H(GR(T)) - тоже связанный гиперграф.

Допустим, что можно убрать еще некоторые строки $r_1, r_2, \ldots, r_p$ из таблицы GR(T) так, что таблица GR(T) превратится в слабо эквивалентной ей таблице $T_m$. (Конечно, это удаление строк не будеть осуществляться путем применения операций 1 и 2 алгоритма 4.1, а с помощью некоторого содержащего отображения).

Пусть можно убрать из GR(T) строку r, которая является крайней строкой для GR(T). Тогда лемма 4.6 показывает, что r содержит изолированную свободную переменную или изолированную

константу. Если r содержит изолированную свободную переменную a, то тогда $T_m$ уже не является таблицей - потому что резюме $T_m$ содержит свободную переменную a, которая не участвует в строках таблицы $T_m$. Следовательно, из GR(T) нельзя убрать крайнюю строку с изолированной свободной переменной. Если r содержит изолированную константу c, то тогда таблица $T_m$ не является слабо эквивалентной таблице GR(T) - таблица $T_m$ содержит меньше констант, чем GR(T).

Допуская, что можно убрать крайние строки из GR(T), получили противоречие. Следовательно, нельзя убрать крайние строки из GR(T).

Допустим, что можно убрать строку r из GR(T) и r не является крайней. Тогда полученная таблица $T_m$ содержит все изолированные свободные переменные и константы из GR(T), но ей соответствует некоторый несвязанный запрос с несвязанным гиперграфом (так как строка r - не крайняя). Получили противоречие, потому что связанный запрос не может быть слабо эквивалентным несвязанному запросу.

Следовательно, если данной таблице T сопоставляется связанный гиперграф, то тогда GR(T) - таблица с минимальным числом строк, слабо эквивалентная данной таблице T.

б) Пусть H(T) - несвязанный гиперграф.

Тогда $H(T) = H_1 \cup H_2 \cup \ldots \cup H_s$, где $H_i$ - связанные гиперграфы для $1 \le i \le s$.

Для каждого гиперграфа $H_i$, $1 \le i \le s$, можно применить пункт а). Тогда видно, что из каждого гиперграфа $H_i$, $1 \le i \le s$, нельзя убрать ребра.

Следовательно, из GR(T) нельзя убрать строки и GR(T) является таблицей с минимальным числом строк, слабо эквивалентной данной таблице T. □

Теорема 4.3 позволяет предложить следующий алгоритм оптимизации таблиц в полиномиальном периоде времени:

АЛГОРИТМ 4.2. Оптимизация данной ациклической таблицы T относительно слабой эквивалентности.

ВХОД: Таблица T, построенная по алгоритму 3.1.

МЕТОД:

1. Применять операцию 1 и операцию 2 алгоритма 4.1 в произвольном порядке, сколько раз возможно.

2. Если уже невозможно больше применять шаг 1, сохранить полученную таблицу GR(T) - она выступает кандидатом минимальной таблицы, слабо эквивалентной входной таблице T.

3. Модифицировать операцию 1 алгоритма 4.1 - можно зачеркнуть все виды однократно встречающихся символов в таблице T, а не только связанные переменные. Применять алгоритм 4.1 над GR(T), причем вместо операции 1 использовать ее модификацию.

ВЫХОД: Пустая запись - когда T является ациклической таблицей. (Тогда GR(T) - ее минимальная таблица). Непустая запись - когда T является циклической таблицей. □

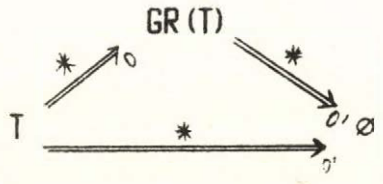
Видно, что алгоритм 4.2 всегда заканчивает работу в полиномиальном периоде времени (как и алгоритмы 2.1 и 4.1).

Для алгоритма 4.2 мы ввели модификацию операции 1 алгоритма 4.1, которую ниже будем называть операцией 1′ и обозначать через $\rightarrow_{N'}$.

Лемма 4.2 показала, что множество построенных по алгоритму

3.1 таблиц и трансформации $\rightarrow_N$ и $\rightarrow_E$ образуют конечную систему Черча-Россера. Аналогичным образом видно, что множество построенных по алгоритму 3.1 таблиц и трансформации $\rightarrow_{N'}$ и $\rightarrow_E$ также являются конечной системой Черча-Россера.

Можно поставить вопрос: всегда ли на выходе алгоритма 4.2 получается пустая запись для ациклических таблиц? Другими словами, имеем ли право в процессе обработки ациклических таблиц зачеркивать сначала только связанные переменные, а затем все остальные символы, которые встречаются только один раз в таблице? Ответ утвердительный, так как $(\rightarrow_N, \rightarrow_E)$ и $(\rightarrow_{N'}, \rightarrow_E)$ образуют конечные системы Черча-Россера вместе с множеством построенных по алгоритму 3.1 таблиц. Пусть через $\overset{*}{\rightarrow_{O'}}$ обозначим многократное применение в произвольном порядке операций 1′ и 2. Тогда из вышесказанного становится ясным, что мы можем редуцировать ациклическую таблицу T до пустой записи следуя любому из обоих путей следующей диаграммы:




Отметим, что в процессе редукции $T \overset{*}{\rightarrow_{O'}} GR(T)$ всегда получаются таблицы, которые являются слабо эквивалентными таблице T. Конечно, в процессе редукции $GR(T) \overset{*}{\rightarrow_{O'}} \varnothing$ и $T \overset{*}{\rightarrow_{O'}} \varnothing$, это неверно.

ПРИМЕР 4.4. Применяя алгоритм 4.2 над таблицей (4.1) видно, что минимальной слабо эквивалентной таблицей является

| A | B | C | D |
|---|---|---|---|
| a | | | |
| a | | $c_1$ | |

□

Запрос (4.1) можно задать и как циклический:

| A | B | C | D | |
|---|---|---|---|---|
| a | | | | |
| a | $b_1$ | | | (AB) |
| | $b_1$ | $b_2$ | | (BC) |
| | | $b_2$ | $c_1$ | (CD) |
| a | | | $c_1$ | (AD) |

(4.4)

В случае слабой эквивалентности запросы (4.1) и (4.4) - эквивалентны, так как между обеими таблицами существуют соответствующие содержащие отображения; но как видно, (4.4) нельзя редуцировать по алгоритму 4.2.

Запросы (4.1) и (4.4) показывают, что время минимизации эквивалентных запросов может зависеть от способа формулировки запроса.

Рассматривая минимизацию запросов в терминах графического представления в определении 3.4, замечаем, что минимальная таблица соответствует "минимальному" графу, содержащему все свободные переменные, константы и разные пути доступа между ними.

Отметим, что алгоритм 4.1 всегда редуцирует только путь доступа данного запроса. Множества $T_L$ и $T_{con}$ запроса сохраняются в процессе оптимизации запроса.

## 4.3. Слабая эквивалентность и оптимизация конъюнктивных циклических запросов

Лемма 4.6 показывает, что если GR(T) - ациклическая таблица и H(GR(T)) - связанный гиперграф, то тогда все крайние строки из GR(T) содержат изолированные свободные переменные или изолированные константы.

Пример 4.3 показывает, что если GR(T) - циклическая таблица, в ней могут участвовать крайние строки без изолированных свободных переменных и констант.

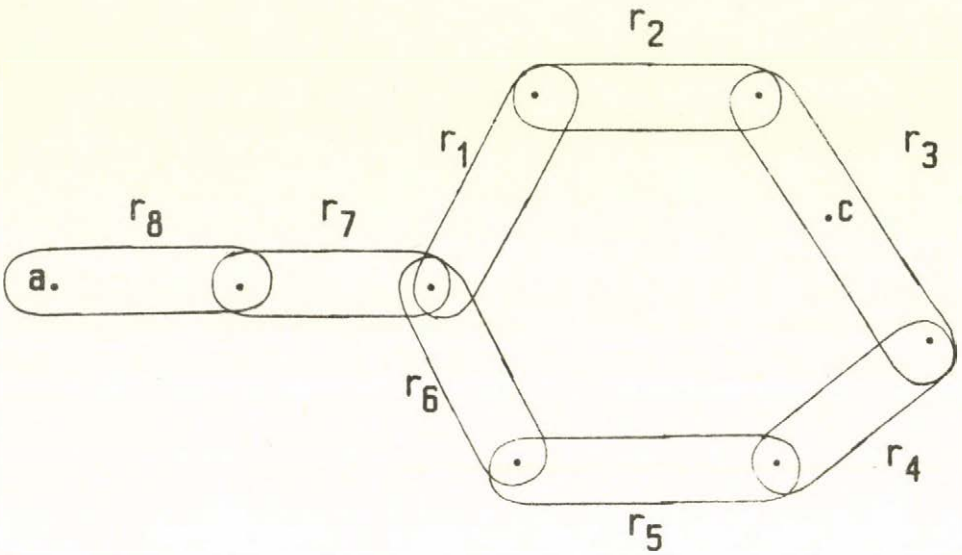
Как известно, циклические гиперграфы всегда содержат циклические компоненты (см. определения 2.11).

ЛЕММА 4.7. Пусть GR(T) - циклическая таблица. Пусть r - крайняя строка из GR(T), которая не содержит изолированных свободных переменных или изолированных констант. Тогда r участвует в некоторой циклической компоненте для GR(T).

ДОКАЗАТЕЛЬСТВО. Ребро e(r) состоит только из связанных вершин и con(e(r))-e(r). Ребро e(r) не содержится целиком в другом ребре гиперграфа H(GR(T)), потому что гиперграф H(GR(T)) - нередуцируемый. Следовательно, в e(r) существуют две вершины $n_1$ и $n_2$ такие,что $n_1 \in e_1 \cap e(r)$ и $n_2 \in e_2 \cap e(r)$, где $e_1$ и $e_2$ - ребра гиперграфа H(GR(T)). Тогда очевидно ребра $e_1$ и $e_2$ связаны с помощью ребра e(r). Но как сказано выше, r является крайней строкой и после ее удаления полученный из H(GR(T)) гиперграф H' продолжает быть связанным. (H' содержит все ребра из H(GR(T)) кроме ребра e(r) ). Тогда ребра $e_1$ и $e_2$ связаны в H' с помощью некоторой цепочки ребер $e_1-e'_1,e'_2,\ldots,e'_{k-1},e'_k-e_2$, где $e'_i \in H'$

и $e'_l \neq e(r)$ для $1 \leq l \leq k$. Тогда очевидно ребра $(e_1, e'_2, \ldots, e'_{k-1}, e_2, e, e_1)$ образуют циклическую компоненту в $H(GR(T))$. □

ПРИМЕР 4.4. Рассмотрим следующий гиперграф для некоторой нередуцируемой таблицы GR(T):




Видно, что в таблице GR(T) строки $r_1$-$r_6$ и $r_8$ являются крайними, а строка $r_7$ не является крайней (потому что удаление строки $r_7$ нарушает связанность гиперграфа). Строка $r_8$ не участвует в циклической компоненте $\{r_1, r_2, r_3, r_4, r_5, r_6\}$ и содержит изолированную свободную переменную a. Строка $r_3$ содержит изолированную константу c. □

Как видно из леммы 2.2, если T - циклическая таблица, то GR(T) тоже является циклической таблицей. Легко видет, что если GR(T) - циклическая таблица и H(GR(T)) - связанный гиперграф, то тогда H(GR(T)) состоит из:

а) циклических компонентов. Эти циклические компоненты могут быть связанными между собой. Их крайные строки могут содержать

изолированные свободные переменные или изолированные константы. (На примере 4.4 крайняя строка $r_3$ содержит изолированную константу, а крайние строки $r_1, r_2, r_4, r_5$ и $r_6$ из циклический компоненты не содержат изолированных свободных переменных и констант).

б) ациклических компонентов, связанных с циклическими компонентами (как $r_7$ и $r_8$ на примере 4.4). Крайные строки ациклических компонентов должни содержать изолированные свободные переменные или константы.

Лемма 4.6 показывает, что ациклические нередуцируемые таблицы состоят из:

а) строк, которые не являются крайными;

б) крайных строк, которые должны содержать изолированные свободные переменные или изолированные константы.

Лемма 4.7 и пример 4.4 показывают, что циклические нередуцируемые таблицы состоят из:

а) строк, которые не являются крайными;

б) крайных строк, которые не участвуют в циклических компонентов и должны содержать изолированные свободные переменые или изолированные константы;

в) крайных строк, которые участвуют в циклических компонентах и могут содержать изолированные свободные переменные или изолированные константы.

В параграфе 4.2 показано, что из ациклической таблице GR(T) нельзя убрать строки из пункта а) - иначе потеряем связанность запроса. Тоже показано, что из ациклической таблице GR(T) нельзя убрать строки из пункта б) - иначе потеряем "смысл" запроса. Увидим, что в случае циклических нередуцируемых таблиц возможно в процессе оптимизации убирать и строки из пунктов а) и б). Сначала рассмотрим строки из пункта в).

ЛЕММА 4.8. Пусть GR(T) - циклическая таблица. Пусть GR(T) содержить циклический компонент $\{r_1,r_2,\ldots,r_p\}$. Пусть $r_i \in \{r_1,r_2,\ldots,r_p\}$ и пусть $r_i$ содержит только свободные переменные и константы. Пусть $T_m$ - таблица, полученная из GR(T) после удаления строки $r_i$. Тогда таблица $T_m$ не может быть слабо эквивалентной таблице GR(T).

ДОКАЗАТЕЛЬСТВО. Допустим, что существует таблица $T_m$ такая, что $T_m$ не содержит некоторую строку $r_i$ из циклического компонента $\{r_1,r_2,\ldots,r_p\}$ и $T_m \sim_w GR(T)$. Тогда между символами GR(T) и $T_m$ существуют соответствующие содержащие отображения $h_1$ и $h_2$ такие, что $h_1: GR(T) \rightarrow T_m$ и $h_2: T_m \rightarrow GR(T)$. Пусть $h_1(r_i)=r_j$, где $r_j \in T_m$. Так как таблица $T_m$ получена из таблицы GR(T) с помощью зачеркивания некоторой ее строки, ясно, что $r_j \in GR(T)$. Из $h_1(r_i)=r_j$ следует, что строка $r_j$ содержит все свободные переменные и константы строки $r_i$. Но как сказано в условии, строка $r_i$ содержит только свободные переменные и константы. Следовательно, символы строки $r_i$ целиком содержатся в символах строки $r_j$ и следовательно, можно применить операцию 2 алгоритма 4.1. Следовательно, GR(T) не является нередуцируемой таблицей - что противоречит условию леммы.

Следовательно, из циклического компонента нередуцируемых циклических таблиц нельзя убирать строки, содержащие только свободные переменные и константы. □

Рассмотрим строки циклических компонентов, содержащие

связанные переменные.

Пусть r - строка некоторой таблицы; обозначим через AC(r) множество всех свободных переменных и констант строки r.

ЛЕММА 4.9. Пусть GR(T) - циклическая нередуцируемая таблица, которая содержит циклический компонент $\{r_1,r_2,\ldots,r_p\}$. Пусть в циклической компоненте $\{r_1,r_2,\ldots,r_p\}$ существуют разные строки $r_1,r_2,\ldots,r_l,r_{l+1}$ и $r_j$ такие, что:

1) $\{b_1^1,b_2^1,\ldots,b_{s1}^1\}\subset r_1\cap r_2$ - все связанные переменные строк $r_1$ и $r_2$;

$\{b_1^2,b_2^2,\ldots,b_{s2}^2\}\subset r_2\cap r_3$ - все связанные переменные строк $r_2$ и $r_3$;

...

$\{b_1^l,b_2^l,\ldots,b_{sl}^l\}\subset r_l\cap r_{l+1}$ - все связанные переменные строк $r_l$ и $r_{l+1}$;

2) $b_m^k\in r_k\cap r_{k+1}$, $1\le k\le l$, $1\le m\le s_k$ и никакой другой строке $r_t\in GR(T)$;

3) $AC(r_1)\cup AC(r_2)\cup\ldots\cup AC(r_l)\cup AC(r_{l+1})\subset AC(r_j)$.

Тогда строки $r_1,r_2,\ldots,r_l,r_{l+1}$ можно убрать из таблице GR(T). Полученная таблица $T_m$ является слабо эквивалентной таблице GR(T) - т.е., $T_m\text{--}_w GR(T)$.

ДОКАЗАТЕЛЬСТВО. Все строки таблицы $T_m$ содержатся в таблице GR(T). Поэтому сразу можно показать содержащее отображение $h_1:T_m\to GR(T)$ - содержащее отображение $h_1$ сохраняет все символы.

Построим содержащее отображение $h_2:GR(T)\to T_m$. $h_2$ сохраняет все свободные переменные и константы и отображает все строки $\{r_1,r_2,\ldots,r_i,r_{i+1}\}$ в строку $r_j$. Пусть символы $\{b_1^1,b_2^1,\ldots,b_{s1}^1\}\subset r_1\cap r_2$ встречаются в строках $r_1$ и $r_2$ в столбцах соответствующих некоторым атрибутам $\{A_1^1,A_2^1,\ldots,A_{s1}^1\}$; тогда $h_2$ отображает символов $\{b_1^1,b_2^1,\ldots,b_{s1}^1\}$ в символы, находящиеся в строке $r_j$ в столбцах, соответствующих атрибутам $\{A_1^1,A_2^1,\ldots,A_{s1}^1\}$. Символы $\{b_1^2,b_2^2,\ldots,b_{s2}^2\}$ отображаются в символы, находящиеся в строке $r_j$ в столбцах атрибутов $\{A_1^2,A_2^2,\ldots,A_{s2}^2\}$ и так далее. Символы $\{b_1^i,b_2^i,\ldots,b_{si}^i\}$ отображаются в символы, находящиеся в строке $r_j$ в столбцах $\{A_1^i,A_2^i,\ldots,A_{si}^i\}$. Так как все вышеупомянутие связанные переменные появляются только два раза в таблице GR(T) и все они отображаются в соответствующие символы из строки $r_j$ таблицы $T_m$, видно, что $h_2:GR(T)\to T_m$.

Из $h_1:T_m\to GR(T)$ и $h_2:GR(T)\to T_m$ следует, что $T_m\text{--}_w GR(T)$. □

Лемма 4.9 показывает, что некоторые нередуцируемые таблицы GR(T) могут не являться минимальными таблицами. Из леммы 4.9 видно, что существуют достаточные условия для дальнейшей оптимизации циклической таблицы GR(T).

Следующая теорема дает нам необходимое и достаточное условие для дальнейшей оптимизации циклической таблицы GR(T).

ТЕОРЕМА 4.4. Пусть GR(T) - циклическая нередуцируемая таблица. Пусть в процессе ее оптимизации из таблицы GR(T) можно

убрать некоторые строки $r_1, r_2, \ldots, r_p$ до получения минимальной таблицы $T_m$. $GR(T) --_w T_m$ тогда и только тогда, когда строки $r_1, r_2, \ldots, r_p$ могут быть сгруппированны в непересекающихся множествах строк $H_1, H_2, \ldots, H_s$ такие, что для каждого множества строк $H_l$, $1 \le l \le s$ существует строка $r_l' \in T_m \cap GR(T)$ и для $H_l$ и $r_l'$ исполнено:

а) $AC(H_l) \subset AC(r_l')$;

б) Все связанные переменные из $H_l$ участвуют в других строках таблицы $T_m$ только тогда, когда участвуют в строке $r_l'$ .

ДОКАЗАТЕЛЬСТВО.

Достаточность. Пусть из таблице $GR(T)$ можно убрать некоторые строки $r_1, r_2, \ldots, r_p$ до получения некоторой таблице $T_m$ и пусть строки $r_1, r_2, \ldots, r_p$ можно сгруппировать в непересекающихся множествах строк $H_1, H_2, \ldots, H_s$ таких, что для каждого множества строк $H_l$, $1 \le l \le s$ существует строка $r_l' \in T_m \cap GR(T)$ и для $H_l$ и $r_l'$ исполнено:

а) $AC(H_l) \subset AC(r_l')$;

б) Все связанные переменные из $H_l$ участвуют в других строках таблицы $T_m$ только тогда, когда участвуют в строке $r_l'$. (Другими словами, если некоторые связанные переменные из $H_l$ не участвуют в строке $r_l'$, тогда они не участвуют и в других строках таблицы $T_m$).

Докажем, что $GR(T) --_w T_m$.

Построим содержащие отображения $h_1 : GR(T) \to T_m$ и $h_2 : T_m \to GR(T)$. Сразу видно, что $T_m \subset GR(T)$. Следовательно, если $h_2$ - содержащее

отображение, которое сохраняет все строки, тогда $h_2: T_m \rightarrow GR(T)$.

Знаем, что $GR(T) - T_m = \{r_1, r_2, \ldots, r_p\}$. Тогда пусть $h_1$ - содержащее отображение, которое сохраняет строки из $T_m$. Рассмотрим как $h_1$ должно отображать строки из $\{r_1, r_2, \ldots, r_p\}$. Как дано в условии, строки $\{r_1, r_2, \ldots, r_p\}$ могут быть сгруппированны в непересекающихся связанных множествах $H_1, H_2, \ldots, H_s$. Для каждого множества $H_l$, $1 \leq l \leq s$ существует соответствующая строка $r_l'$ из таблицы $T_m$. Пусть $h_1(H_l) = r_l'$ для $1 \leq l \leq s$. При этом $h_1$ сохраняет значения всех свободных переменных и констант из $H_l$, потому что $AC(H_l) \subset AC(r_l')$. $h_1$ отображает связанные переменные из $H_l$ в символы, поставленные в строке $r_l'$ в соответствующих столбцах. Здесь нужно рассмотреть два случая:

- Когда связанные переменные из $H_l$ участвуют только в $H_l$ и не участвуют в строке $r_l'$ и в других строках таблицы $T_m$ - тогда $h_1$ не сохраняет эти связанные переменные, а отображает их в символы, поставленые в строке $r_l'$ в соответствующих столбцах. Так как множества строк $H_1, H_2, \ldots, H_s$ не пересекаются, вышеупомянутые связанные переменные встречаются только в $H_l$. Видно, что отображение $h_1$ задано коректно;

- Когда связанные переменные из $H_l$ участвуют в строке $r_l'$. Тогда отображение $h_1$ сохраняет эти связанные переменные, которые могут участвовать и в других строк таблицы $T_m$. Так как $h_1$ сохраняет все строки из $T_m$, видно, что и в этом случае

отображение $h_1$ задано коректно.

Следовательно, существуют содержащие отображения $h_1: GR(T) \to T_m$ и $h_2: T_m \to GR(T)$. Следовательно, $GR(T) --_w T_m$.

Необходимость. Пусть из циклической нередуцируемой таблицы $GR(T)$ можно убрать некоторые строки $r_1, r_2, \ldots, r_p$ так, что полученная таблица $T_m$ является минимальной слабоэквивалентной таблицей для таблицы $GR(T)$, т.е. $GR(T) --_w T_m$.

Докажем, что строки $r_1, r_2, \ldots, r_p$ могут быть сгруппированны в непересекающихся множествах строк $H_1, H_2, \ldots, H_s$ таких, что для каждого множество строк $H_l$, $1 \le l \le s$ существует строка $r_l' \in T_m \cap GR(T)$ и для $H_l$ и $r_l'$ исполнено:

а) $AC(H_l) \subset AC(r_l')$;

б) Все связанные переменные из $H_l$ участвуют в разных строках таблицы $T_m$ только тогда, когда участвуют в строке $r_l'$.

Знаем, что $GR(T) --_w T_m$. Следовательно, существуют содержащие отображения $h_1: GR(T) \to T_m$ и $h_2: T_m \to GR(T)$. Рассмотрим содержащее отображение $h_1$. Так как $T_m \subset GR(T)$, то $h_1$ отображает строк $\{r_1, r_2, \ldots, r_p\} - GR(T) - T_m$ в некоторые строки из $T_m$. Пусть это будут строки $r_1', r_2', \ldots, r_s'$. Обозначим через $H_1$ эти строки из $\{r_1, r_2, \ldots, r_p\}$, которые с помощью $h_1$ отображаются в $r_1'$; обозначим через $H_2$ эти строки из $\{r_1, r_2, \ldots, r_p\}$, которые с помощью $h_1$ отображаются в $r_2'$;...; обозначим через $H_s$ эти строки из $\{r_1, r_2, \ldots, r_p\}$, которые с помощью $h_1$ отображаются в

$r_s'$. Известно, что $h_1$ отображает каждую строку из GR(T) в точно одну строку из $T_m$; поэтому множества строк $H_1, H_2, \ldots, H_s$ не пересекаются.

Отметим, что содержащее отображение $h_1$ сохраняет все строки из $GR(T) \cap T_m$ (иначе, если $h_1$ отображает две строки из $GR(T) \cap T_m$ в одну строку из $T_m$, то из таблицы $T_m$ можно убрать еще некоторые строки и она не являлась бы минимальной таблицей для GR(T) ). Следовательно, $h_1$ отображает все символы из $GR(T) \cap T_m$ в их самых.

Так как $h_1$ отображает строки из множества $H_l, 1 \le l \le s$ в строку $r_l'$ из $T_m \cap GR(T)$, видно, что $AC(H_l) \subset AC(r_l')$ для $1 \le l \le s$.

Легко видеть, что множества $H_l$, $1 \le l \le s$ не могут состоять только из строк, в которых не участвуют связанные переменные (а только свободные переменные и константы). Лемма 4.8 показала, что из GR(T) нельзя убрать строку, содержащую только свободные переменные и константы. Следовательно, в множествах $H_l$, $1 \le l \le s$, должни участвовать некоторые связанные переменные.

Знаем, что для $1 \le l \le s$, $r_l' \in T_m \cap GR(T)$ и поэтому $h_1(r_l') = r_l'$. Если некоторая связанная переменная $b_1$ из множества $H_l$ встречается в строках таблицы $T_m$, тогда $b_1$ должна встречаться и в строке $r_l'$ в соответствующем столбце и тогда $h_1(b_1) = b_1$. Если некоторая связанная переменная $b_2$ из $H_l$ не участвует в строке $r_l'$, то тогда $h_1(b_2) = s$, где s - символ, который стоит в строке $r_l'$ в этом столбце, в котором встречается в $H_l$ связанная переменная $b_2$. □

Теорема 4.4 позволяет нам предложить алгоритм для дальнейшей оптимизации циклических нередуцируемых таблиц. Алгоритм начинает работу над таблицей GR(T), которая является выходом алгоритма 4.2. Знаем, что GR(T) содержит циклическую компоненту, которая может не участвовать в минимальной таблице $T_m$.

Алгоритм работает на основе теоремы 4.4 - для каждой строки $r \in GR(T)$ алгоритм проверяет возможно ли построить множество строк $H_i$ такое, что для некоторого содержащего отображения $h_1$ имеем: $h_1(H_i)-r$. Нужно отметить, что отдельные множества $H_1, H_2, \ldots, H_s$ в теореме 4.5 не пересекаются; поэтому, если алгоритм найдет некоторое множество $H_i$, после его удаления из таблицы GR(T) не надо проверять сначала все строки из таблицы GR(T), а можно продолжать поиск другого множества $H_{i+1}$.

Введем следующие обозначения: Строки из GR(T) будем обозначать через $r_1, r_2, \ldots, r_n$. Через H будем обозначать множество строк, которые можно удалить из GR(T). Если M - множество строк, то через V(M) будем обозначать множество символов всех строк из M.

АЛГОРИТМ 4.3. Оптимизация циклической нередуцируемой таблицы

ВХОД: Циклическая нередуцируемая таблица GR(T), получена с помощью алгоритма 4.2.

МЕТОД:

1. GR'(T):-GR(T); i:-1.
2. Если i>n, то $T_m$:-GR(T). Конец работы.
3. Для строки $r_i \in GR'(T)$, проверяется участвует ли она еще в

таблице GR(T) (при удалении множества строк H можно убрать строку с номером больше, чем у текущей строки). Если $r_I$ уже удалена, тогда I:-I+1, перейти к пункту 2.

4. Для строки $r_I \in GR(T)$ построить множество S, которое содержит все строки из GR(T), имеющие общие символы с $r_I$. Пусть $S-\{h_1^I, h_2^I, \ldots, h_t^I\}$.

5. Пусть J:-1; S':-S.

6. Если J>t, I:-I+1, перейти к пункту 2.

7. Для строки $h_J^I \in S'$ проверяется участвует ли она еще в множестве S (потому что при удалении множества строк H можно убрать строку с номером больше, чем у текущей строки). Если $h_J^I$ уже удалена, тогда J:-J+1 и перейти к пункту 6.

8. Пусть $h:-h_J^I$, H-∅; V'(H):-∅.

9. Пусть H:-H∪{h} и V'(H):-V'(H) ∪ (V(h)-V($r_I$)).

10. Если V'(H) содержит свободные переменные или константы, строки H не могут быть удалены из GR(T). Тогда $H:-H-\{h_J^I\}$, J:-J+1, перейти к пункту 6.

11. Для символа $v \in V'(H)$ найти строку $h \in GR(T)-(\{r_I\} \cup H)$, которая содержит v.

12. Если существует такая строка h, перейти к пункту 9.

13. Если не существует такая строка h, V'(H):-V'(H)-{v}.

14. Если V'(H)-∅, тогда GR(T):-GR(T)-H, J:-J+1, S:-S-H и перейти к пункту 6. Если $V'(H) \neq \varnothing$, перейти к пункту 11.

ВЫХОД: Таблица $T_{III}$, которая является минимальной слабоэквивалентной таблицей для входной таблицы GR(T). □

С помощью процедуры

```
procedure include(h,r)
V'(H) := V'(H) ∪ (v(h) - v(r))
if <V' содержит свободные переменные или константы>
      then V' := ∅, H := ∅
      else H := H ∪ {h}
endif
return
```

алгоритм 4.3 может быть представлен следующим образом:

```
GR'(T) := GR(T)
while GR'(T) ≠ ∅ do
      для r∈GR'(T), S := {r'∈GR'(T) | V(r)∩V(r')≠∅}
      while S≠∅ do
      для h'∈S, H := ∅, V'(H) := ∅
      include (h',r)
            while V'(H) ≠ ∅ do
                  для v∈V'(H)
                  if <(∃h)∈(GR(T)-{r}-H) и v∈V(h)>
                  then include(h,r)
                  else V'(H) := V'(H)-{v}
                  endif
            endwhile
            S := S-{H}, GR(T) := GR(T)-H
      endwhile
      GR'(T) := GR'(T) - {r}
endwhile
```

Видно, что алгоритм 4.3 состоит из четырех циклов. Пусть n - число реляционных схем, а m - число атрибутов универсума. В

первом цикле рассматриваются не больше чем $n^2(n-1)m$ элементов. Во втором цикле рассматриваются не больше чем $((n-1)n)^2m$ элементов, в третьем цикле - не больше чем $nm$ элементов. В четвертом цикле (записан в виде оператора **if...then...else**) рассматриваются не больше чем $(n-2)nm+nm$ элементов. В процедуре Include рассматриваются не больше чем $nm$ элементов.

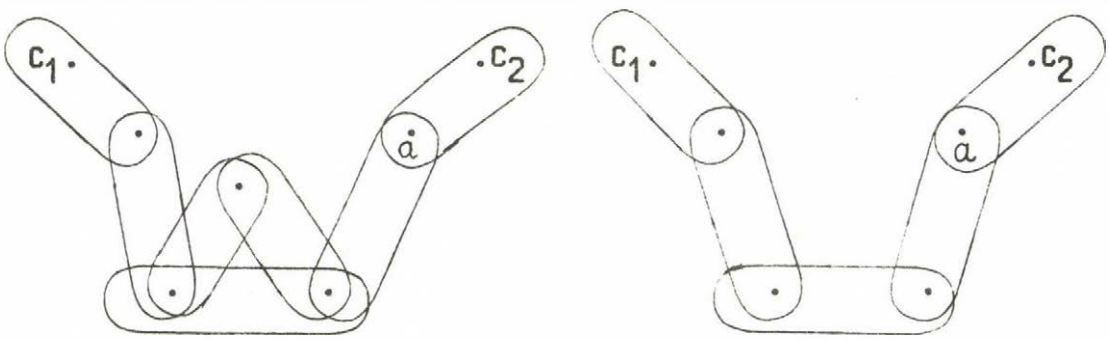
Общую оценку сложности алгоритма 4.3 можно записать следующим образом:

$$n((n-1)m((n-1)nm(nm(nm(n-2)+nm)(n-1)n))n) \leq n^{10}m^4.$$

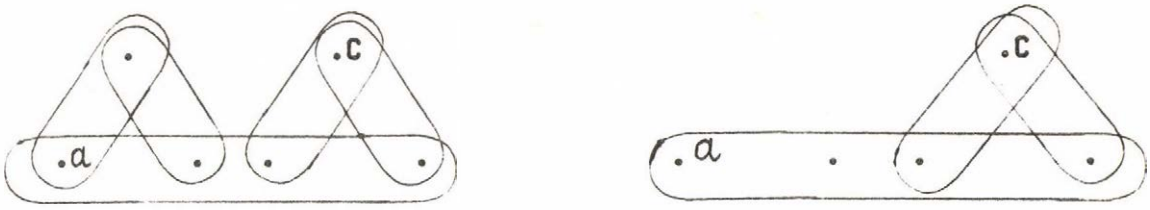
Следовательно, алгоритм 4.3 заканчивает работу в полиномиальном периоде времени.

ПРИМЕР 4.5. Рассмотрим несколько интересных циклических гиперграфов для нередуцируемых таблиц. Даны и гиперграфы, соответствующие минимальным таблицам:

а) гиперграф с одной циклической компонентой, которая не присутствует в минимальной таблице:




б) гиперграф с двумя циклическими компонентами; одна из них может быть устранена из данной таблицы:




□

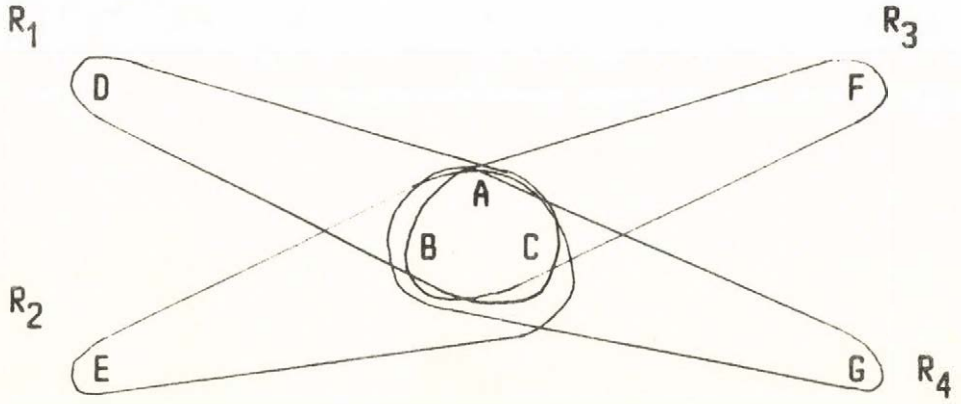
### 4.4. Виды запросов к разным реляционным схемам

В этом параграфе рассмотрим вопрос какие виды запросов можно задавать в разных реляционных схемах. Например, если дана A-ациклическая схема реляционной базы данных, возможно ли в ней сформулировать циклический запрос и т.д.

Рассмотрим следующий пример:

ПРИМЕР 4.6. Пусть даны реляционные схемы $R_1 = \{A,B,C,D\}$, $R_2 = \{A,B,C,E\}$, $R_3 = \{A,B,C,F\}$ и $R_4 = \{A,B,C,G\}$.

Гиперграф схемы $\mathbf{R} = \{R_1, R_2, R_3, R_4\}$ выглядит следующим образом:




(4.5)

Рассмотрим запрос в этой схеме:

RETRIEVE ($R_1$.D) (4.6)

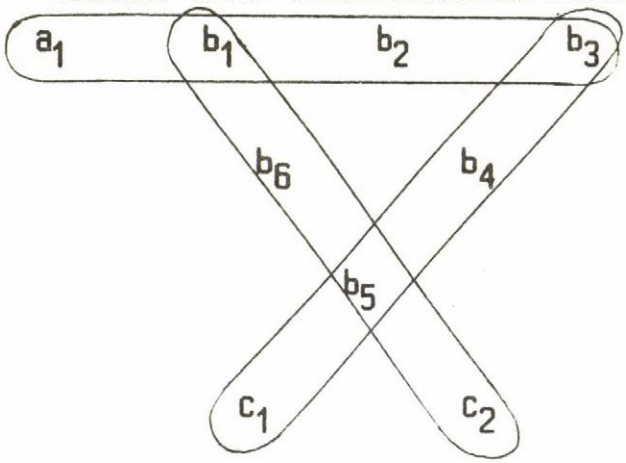
WHERE ($R_2$.E='$c_1$') AND ($R_3$.F='$c_2$')

AND ($R_1$.A=$R_3$.A) AND ($R_2$.B=$R_3$.B) AND ($R_1$.C=$R_2$.C).

Запрос (4.6) имеет таблицу

| A | B | C | D | E | F | G | (4.7) |
|---|---|---|---|---|---|---|---|
| | | | $a_1$ | | | | |
| $b_1$ | $b_2$ | $b_3$ | $a_1$ | | | | ($R_1$) |
| $b_4$ | $b_5$ | $b_3$ | | $c_1$ | | | ($R_2$) |
| $b_1$ | $b_5$ | $b_6$ | | | $c_2$ | | ($R_3$) |

Таблице (4.7) сопоставляется гиперграф




(4.8)

Гиперграф (4.8) - циклический, а схема $\mathbf{R}$ - A-ациклическая.

Этот пример показывает, что в A-ациклических схемах можно формулировать циклические запросы. □

ПРИМЕР 4.7. В схеме (4.5) можно задать и A-ациклический запрос:

| A | B | C | D | E | F | G | |
|---|---|---|---|---|---|---|---|
| | | $a_1$ | | | | | |
| $b_1$ | $b_2$ | $a_1$ | | $b_3$ | | | $(R_1)$ |
| $b_4$ | $b_5$ | $a_1$ | $b_7$ | | | | $(R_2)$ |
| $b_1$ | $b_5$ | $b_6$ | | | $b_8$ | | $(R_3)$ |
| $b_1$ | $b_5$ | $a_1$ | | | | $b_9$ | $(R_4)$ |

Эта таблица показывает, что в A-ациклических схемах можно задавать и A-ациклические запросы. □

ОПРЕДЕЛЕНИЕ 4.16. Пусть H - гиперграф, который содержит цепочку ребер и вершин

$(e_1, x_1, e_2, x_2, \ldots, e_m, x_m, e_{m+1})$

такая, что

1) $e_1, e_2, \ldots, e_m$ - разные ребра гиперграфа;

2) $x_1, x_2, \ldots, x_m$ - разные вершины гиперграфа;

3) $m \geq 3$;

4) $e_{m+1} = e_1$;

5) $x_i \in e_i \cap e_{i+1}$ для $1 \leq i \leq m$.

Тогда гиперграф H будем называть D-циклическим. □

ОПРЕДЕЛЕНИЕ 4.17. Пусть H - гиперграф, который не является D-циклическим. Тогда H будем называть D-ациклическим. □

ТЕОРЕМА 4.4. Пусть **R** - схема реляционной базы данных, которой соответствует D-ациклический гиперграф. Тогда в **R** можно задавать только запросы из **K1** с D-ациклическими гиперграфами.

ДОКАЗАТЕЛЬСТВО. Обозначим через H гиперграф, соответствующий

схеме $R$.

Допустим, что в $R$ можно задать некоторый D-циклический запрос $q$. Обозначим через T таблицу запроса $q$, построенную по алгоритму 3.1. Тогда в таблице T существует множество строк $S_1, S_2, \ldots, S_m, S_{m+1}$ так, что

1) $m \geq 3$;

2) $S_{m+1} - S_1$;

3) строки $S_1$ и $S_2$ имеют общий символ (переменная или константа) $x_1$; строки $S_2$ и $S_3$ имеют общий символ (переменная или константа) $x_2$;...; строки $S_m$ и $S_{m+1}$ имеют общий символ (переменая или константа) $x_m$.

4) $x_i \in S_i \cap S_{i+1}$, $1 \leq i \leq m$ .

Пусть строка $S_1$ имеет маркер $R_1$, строка $S_2$ имеет маркер $R_2$,..., строка $S_m$ имеет маркер $R_m$. Здесь $R_i \in R$ для $1 \leq i \leq m$. Общий для $S_1$ и $S_2$ символ $x_1$ поставлен в таблице T в столбце некоторого атрибута $A_1$, где $A_1 \in R_1 \cap R_2$; общий для $S_2$ и $S_3$ символ $x_2$ поставлен в таблице T в столбце некоторого атрибута $A_2$, где $A_2 \in R_2 \cap R_3$;...; общий для $S_m$ и $S_{m+1}$ символ $x_m$ поставлен в таблице T в столбце некоторого атрибута $A_m$, где $A_m \in R_m \cap R_{m+1}$. Следовательно, можно построить цепочку реляционных схем и их атрибутов:

$$(R_1, A_1, R_2, A_2, \ldots, R_m, A_m, R_{m+1}) \qquad (4.9)$$

такая, что:

1) $m \geq 3$;

2) $R_{m+1} - R_1$;

3) $A_i \in R_i \cap R_{i+1}$ для $1 \leq i \leq m$.

Исследуя цепочку (4.9) и соответствующую цепочку вершин и ребер гиперграфа H, легко видеть, что H является D-циклическим гиперграфом. Это противоречить условию теоремы.

Следовательно, в схемах с D-ациклическими гиперграфами можно формулировать только запросы с D-ациклическими гиперграфами.

□

В [Fag83] введены разные виды ациклических гиперграфов. Следуя определения в [Fag83], не трудно видить, что определенные нами D-ациклические гиперграфы можно изобразить следующим образом:


Можно поставить следующий вопрос: Пусть u - вид запрос из K1 и v - вид схема: возможно ли в схеме вида v сформулировать запрос вида u?

Примеры 4.6, 4.7 и теорема 4.4 дают ответ на этот вопрос в следующих случаях:

| вид схемы / вид запрос | циклическая | A-ациклическая | D-ациклическая |
|---|---|---|---|
| циклический запрос | | да | нет |
| A-ациклический запрос | | да | нет |
| D-ациклический запрос | | | да |

Легко видеть, что для введеных в этой работе видов схем, эту таблицу можно заполнить следующим образом:

| вид схемы / вид запрос | циклическая | A-ациклическая | D-ациклическая |
|---|---|---|---|
| циклический запрос | да | да | нет |
| A-ациклический запрос | да | да | нет |
| D-ациклический запрос | да | да | да |

# Г Л А В А  П Я Т А Я

# УСЛОВИЯ ЭКВИВАЛЕНТНОСТИ ЗАПРОСОВ

Как было отмечено в введении и в параграфе 1.5, проблема проверки эквивалентности двух заданных реляционных выражений является в общем случае алгоритмически неразрешимой, или, для некоторых классов выражений, NP-полной. В отдельных случаях существуют алгоритмы проверки эквивалентности и оптимизации в полиномиальном периоде времени. Обсудим общее направление исследования проблемы эквивалентности и оптимизации реляционных выражений.

Первое основное замечание, которое может быть сделано - это то, что до сих пор при исследовании проблемы эквивалентности запросов не учитывались свойства схемы базы данных, к которой заданы конкретные запросы. Как показывают многочисленные подробные примеры в параграфе 1.5, при построении таблиц SPJ-выражений рассматриваются только связи между атрибутами, вытекающие из синтаксиса и семантики конкретного представления запроса. Но кроме описанных в запросе связей между атрибутами, существуют и другие связи между атрибутами - а именно, связи заданные с помощью свойств конкретной схемы базы данных. Эти связи не рассматривались до сих пор при изучении проблемы эквивалентности запросов несмотря на то, что каждый реляционный

запрос всегда задается к конкретной базе данных (а не вообще). По нашему мнению, такая общая постановка проблемы эквивалентности и оптимизации запросов не приведет к более эффективным алгоритмам проверки эквивалентности и оптимизации.

Второе основное замечание, которое может быть сделано - это то, что до сих пор при исследовании проблемы эквивалентности запросов не рассматривались и ограничения над "количеством" информации, которое присутствует в состоянии базы данных в тот момент, когда задается запрос к ней. Именно это присутствие (или отсутствие) некоторых значений отдельных атрибутов в отношениях может сделать (или не сделать) два запроса эквивалентными.

Наши условия эквивалентности запросов ([An986] и [An987a]) будут сформулированы с учетом

1) свойств схемы реляционной базы данных;

2) "количества" информации, находящейся в текущем состоянии этой базы данных.

Таким образом, эквивалентность не является свойством только запросов; эквивалентность является свойством схемы базы данных и таким образом, эквивалентность запросов вытекает и из семантики действительности.

В [An986] дано достаточное условие эквивалентности запросов, заданных в некоторой циклической схеме базы данных. Показано, что если состояние базы данных является сплошь-совпадающим и если некоторые конъюнктивные запросы $q_1$ и $q_2$ имеют пути доступа определенного вида, то тогда $q_1 \equiv_w q_2$. Здесь остановимся на возможности проверки эквивалентности запросов в ациклических схемах баз данных. Наша цель - найти необходимые и достаточные условия эквивалентности запросов. В параграфе 5.2 дано необходимое и достаточное условие эквивалентности запросов в D-

ациклических схемах. Этот результат и достаточное условие в [An986] показывают, что возможно искать условия эквивалентности запросов, которые учитывают структуру запросов и которые сформулированы для разных видов реляционных схем - для циклических, D-ациклических и так далее. Результат в параграфе 5.2 и достаточное условие в [An986] показывают, что структура эквивалентных запросов зависит от структуры схемы базы данных, к которой адресованы данные запросы.

Как и в главе 4, будем рассматривать только коньюнктивные запросы на языке QUEL с одним RANGE-оператором для каждой реляционной схемы, участвующей в описании запроса. Запросы этого вида записываются следующим образом:

$q$: RETRIEVE $(R_{11}.A_{11}, R_{12}.A_{12}, \ldots, R_{1s}.A_{1s})$

WHERE $(R_{J1}.A_{J1} \theta c_1)$ AND $(R_{J2}.A_{J2} \theta c_2)$ AND ... AND $(R_{Jp}.A_{Jp} \theta c_p)$

AND $(R_{k1}.A_{k1} \theta R_{k2}.A_{k1})$ AND $(R_{k3}.A_{k3} \theta R_{k4}.A_{k3})$ AND ...

AND $(R_{kt}.A_{kt} \theta R_{k(t+1)}.A_{kt})$ .

В главе 3 мы ввели представление этих запросов в виде упорядоченной четверки (см. определения 3.5).

Пусть $q_1$ и $q_2$ - запросы на QUEL, которые имеют следующие представления в виде упорядоченной четверки:

$q_1 = (T_L^1, T_{con}^1, T_{rr}^1, \Sigma^1)$ и

$q_2 = (T_L^2, T_{con}^2, T_{rr}^2, \Sigma^2)$.

Легко видеть, что необходимыми условиями для слабой и сильной эквивалентности запросов $q_1$ и $q_2$ являются условия:

$T_L^1 = T_L^2$,

$T_{con}^1 = T_{con}^2$ и

$\Sigma^1$ эквивалентно $\Sigma^2$, т.е. сравнения из $\Sigma^1$ либо одинаковые со сравнениями из $\Sigma^2$, либо следуют из них.

Следовательно, эквивалентные запросы могут отличаться только разными путями доступа.

Дальше будем рассматривать те запросы, для которых множество ограничений $\Sigma$ - пустое, т.е. $\Sigma = \varnothing$.

ОПРЕДЕЛЕНИЕ 5.1. Пусть $q = (T_L, T_{con}, T_{rr}, \varnothing)$ - запрос с связанным путем доступа:

П: $(R_1.A_1 = R_2.A_1)$ AND $(R_2.A_2 = R_3.A_2)$ AND...

AND $(R_t.A_t = R_{t+1}.A_t)$

Пусть условия Р:

Р: $(R_{l1}.A_{l1} = R_{l2}.A_{l1})$ AND ... AND $(R_{lk}.A_{lk} = R_{l(k+1)}.A_{lk})$

участвуют в пути доступа П, т.е. $P \subset П$. Условия Р будем называть <u>развлетвлением</u> пути доступа П тогда и только тогда, когда:

1) В условиях П-Р участвуют имена всех атрибутов, участвующих в $T_L$ и $T_{con}$;

2) Условия Р - связанные;

3) Условия П-Р являются связанным путем доступа для запроса q. □

ПРИМЕР 5.1. Рассмотрим базу данных из примера 1.4:

ЧАСТЬ (ЧИМЯ,ЧНОМЕР,ЦЕНА)

ПОСТАВЩИК (ПИМЯ,ПНОМЕР,ПАДРЕС,ПГОРОД)

КЛИЕНТ (КИМЯ,КНОМЕР,КАДРЕС,КГОРОД)

ПОСТАВКА (ЧНОМЕР,ПНОМЕР,КНОМЕР,КОЛИЧЕСТВО)

ОБЯЗАННОСТЬ (ЧНОМЕР,ПНОМЕР).

Пусть $q_1$ - запрос

```
q1:  RETRIEVE (ОБЯЗАННОСТЬ.ПНОМЕР)
        WHERE (ЧАСТЬ.ЦЕНА='200')
        AND (ОБЯЗАННОСТЬ.ЧНОМЕР=ЧАСТЬ.ЧНОМЕР).
```

Пусть $q_1'$ - запрос

```
q1': RETRIEVE (ОБЯЗАННОСТЬ.ПНОМЕР)
        WHERE (ЧАСТЬ.ЦЕНА='200')
        AND (ОБЯЗАННОСТЬ.ЧНОМЕР=ЧАСТЬ.ЧНОМЕР)
        AND (ОБЯЗАННОСТЬ.ПНОМЕР=ПОСТАВЩИК.ПНОМЕР)
        AND (ПОСТАВЩИК.ПНОМЕР=ПОСТАВКА.ПНОМЕР)
        AND (ПОСТАВКА.КНОМЕР=КЛИЕНТ.КНОМЕР).
```

Тогда условия

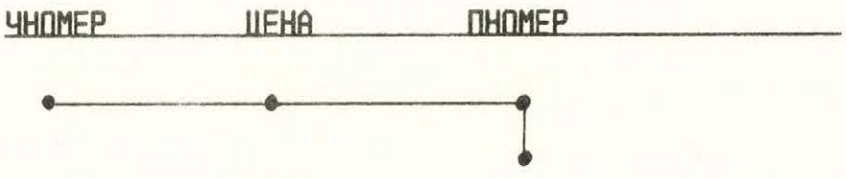
```
           (ОБЯЗАННОСТЬ.ПНОМЕР=ПОСТАВЩИК.ПНОМЕР)
        AND (ПОСТАВЩИК.ПНОМЕР=ПОСТАВКА.ПНОМЕР)
        AND (ПОСТАВКА.КНОМЕР=КЛИЕНТ.КНОМЕР)
```

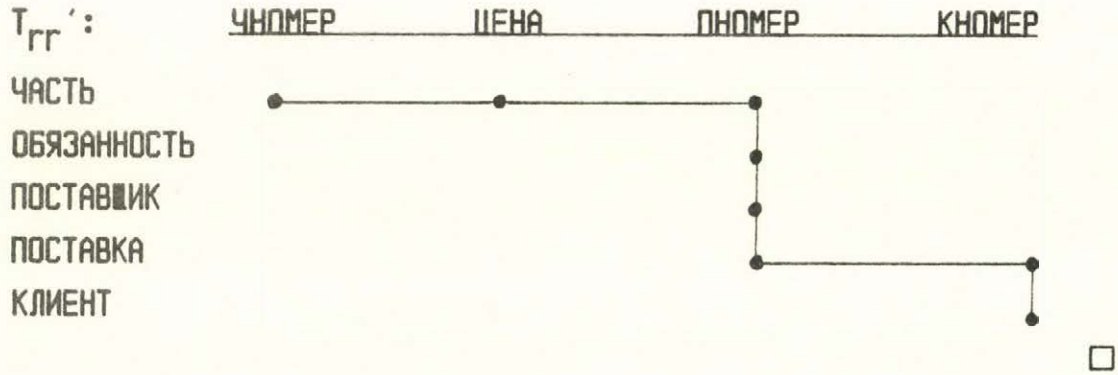
являются разветвлением пути доступа

```
           (ОБЯЗАННОСТЬ.ЧНОМЕР=ЧАСТЬ.ЧНОМЕР)
      AND (ОБЯЗАННОСТЬ.ПНОМЕР=ПОСТАВЩИК.ПНОМЕР)
      AND (ПОСТАВЩИК.ПНОМЕР=ПОСТАВКА.ПНОМЕР)
      AND (ПОСТАВКА.КНОМЕР=КЛИЕНТ.КНОМЕР).
```

Пути доступа запросов $q_1$ и $q_1'$ имеют соответственно графические представления $T_{rr}$ и $T_{rr}'$:

$T_{rr}$:






□

## 5.1. Эквивалентность запросов в D-ациклических схем

Докажем следующую теорему.

ТЕОРЕМА 5.1. Пусть $q_1$ и $q_2$ - запросы с путями доступа соответственно $П_1$ и $П_2$. Пусть $q_1=(T^1_L, T^1_{con}, T^1_{rr}, \varnothing)$, $q_2=(T^2_L, T^2_{con}, T^2_{rr}, \varnothing)$ и $T^1_L=T^2_L$, $T^1_{con}=T^2_{con}$. Пусть $q_1$ и $q_2$ заданы в некоторой D-ациклической схеме **R**. $q_1 \text{--}_w q_2$ тогда и только тогда, когда $П_1$ получен из $П_2$ с помощью добавления и(или) удаления некоторых разветвлений $P_1, P_2, \ldots, P_s$, где графические представления $P_l$ и $П_2$ имеют не больше чем одну общую строку для $1 \le l \le s$.

ДОКАЗАТЕЛЬСТВО.

Достаточность. Пусть $T^1_L=T^2_L$, $T^1_{con}=T^2_{con}$ и $П_1$ получен из $П_2$ с помощью добавления или удаления некоторых непересекающихся разветвлений $P_1, P_2, \ldots, P_s$, где графические представления $P_l$ и $П_2$ имеют не больше чем одну общую строку для $1 \le l \le s$.

Пусть запрос $q_2$ имеет таблицу $T_2$, построенную по алгоритму 3.1 - т.е., в таблице $T_2$ для каждой реляционной схемы из $\mathbf{R}$ участвует не больше чем одна строка, маркированная с именем этой реляционной схемы. Рассмотрим сохранение эквивалентности запросов $q_1$ и $q_2$ при добавлении и устранении разветвлений.

а) Допустим, что к пути доступа $\Pi_2$ можно добавить некоторое разветвление P такое, что графические представления для P и для $\Pi_2$ имеют только одну общую строку. Таким образом, получается новый запрос с путем доступа $\Pi_2 \cup P$. Обозначим через $q_2'$ запрос $(T_L^2,\ T_{con}^2,\ T_{rr'},\ \varnothing)$, где $T_{rr'}$ - графическое представление пути доступа $\Pi_2 \cup P$. Обозначим через $T_2'$ таблицу, полученную для запроса $q_2'$ с помощью алгоритма 3.1. Покажем, что $T_2 \text{--}_w T_2'$.

Сразу видно, что $T_2 \subset T_2'$ - потому что запросы $q_2$ и $q_2'$ имеют одинаковые множества $T_L^2$, $T_{con}^2$ и $\Pi_2 \subset \Pi_2 \cup P$. Следовательно, существует содержащее отображение $h_1 : T_2 \to T_2'$, которое сохраняет все символы.

Пусть $\Pi_2$ - путь доступа

$$\Pi_2:\quad (R_1.A_1 = R_2.A_1)\ \text{AND}\ (R_3.A_3 = R_4.A_3)\ \text{AND}\ \ldots$$
$$\text{AND}\ (R_t.A_t = R_{t+1}.A_t)$$

Пусть условия P:

$$P:\quad (R_{11}.A_{11} = R_{12}.A_{11})\ \text{AND}\ \ldots\ \text{AND}\ (R_{1k}.A_{1k} = R_{1(k+1)}.A_{1k})$$

Графические представления $\Pi_2$ и P имеют только одну общую строку - следовательно, только одна из реляционных схем в P встречается и в пути доступа $\Pi_2$. Пусть это будет реляционная

схема $R_{11}$. Остальные реляционные схемы $R_{12},R_{13},\ldots,R_{1(k+1)}$ не участвуют в $\Pi_2$ (иначе графические представления для P и для $\Pi_2$ имеют больше чем одну общую строку - это следует из способа построения графического представления). Следовательно, таблица $T_2$ содержит строки для реляционных схем $R_1,R_2,\ldots,R_t,R_{t+1}$, а таблица $T_2'$ - строки для реляционных схем $R_1,R_2,\ldots,R_t,R_{t+1},R_{12},R_{13},\ldots,R_{1(k+1)}$. (Здесь предполагается, что реляционная схема $R_{11}=R_j$, где $R_j\in\{R_1,R_2,\ldots,R_t,R_{t+1}\}$.) Покажем, что существует содержащее отображение $h_2$: $T_2'\to T_2$.

Содержащее отображение $h_2$ сохраняет все строки с маркерами $R_1,R_2,\ldots,R_{t+1}$ - как было сказано выше, эти строки таблицы $T_2$ и $T_2'$ являются одинаковыми. В таблице $T_2'$, в строках для $R_{12},R_{13},\ldots,R_{1(k+1)}$, в столбцах для U-A, встречаются только связанные переменные, которые не участвуют в остальных строках таблицы $T_2'$. Следовательно, можем построить содержащее отображение $h_2$, которое отображает все строки для $R_{12},R_{13},\ldots,R_{1(k+1)}$ в строку для $R_j$.

Содержащие отображения $h_1$ и $h_2$ показывают, что $T_2 \equiv_w T_2'$.

б) Допустим, что к пути доступа $\Pi_2$ можно добавить некоторое разветвление P такое, что графические представления для P и для $\Pi_2$ не имеют общих строк. Таким образом, получается новый запрос с путем доступа $\Pi_2 \cup P$. Обозначим через $q_2'$ запрос $(T_L^2, T_{con}^2, T_{rr}', \varnothing)$, где $T_{rr}'$ - графическое представление пути доступа $\Pi_2 \cup P$. Обозначим через $T_2'$ таблицу, полученную для запроса $q_2'$ с

помощью алгоритма 3.1. Покажем, что $T_2 \text{--}_w T_2'$.

Сразу видно, что $T_2 \subset T_2'$ - потому что запросы $q_2$ и $q_2'$ имеют одинаковые множества $T_L^2$, $T_{con}^2$ и $\Pi_2 \subset \Pi_2 \cup P$. Следовательно, существует содержащее отображение $h_1 : T_2 \to T_2'$, которое сохраняет все символы.

Таблица $T_2'$ содержит строки для реляционных схем из разветвления P. В этих строках участвуют только связанные переменные, которые не встречаются в строках для реляционных схем из пути доступа $\Pi_2$. Следовательно, можем построить содержащее отображение $h_2 : T_2' \to T_2$, которое сохраняет все строки для реляционных схем из $\Pi_2$ и отображает все строки для реляционных схем из P в произвольную строку для некоторой реляционной схемы из $\Pi_2$.

в) Допустим, что из пути доступа $\Pi_2$ можно удалить некоторое разветвление P такое, что графические представления для P и для $\Pi_2$ имеют только одну общую строку. Таким образом, получается новый запрос с путем доступа $\Pi_2$-P. Обозначим через $q_2'$ запрос $(T_L^2, T_{con}^2, T_{rr}', \varnothing)$, где $T_{rr}'$ - графическое представление пути доступа $\Pi_2$-P. Обозначим через $T_2'$ таблицу, полученную для запроса $q_2'$ с помощью алгоритма 3.1. Этот случай эквивалентен случаю а), но здесь $T_2'$ имеет меньше строк, чем $T_2$. Поэтому $T_2 \text{--}_w T_2'$.

г) Допустим, что из пути доступа $\Pi_2$ можно удалить некоторое

разветвление P такое, что графические представления для P и для $П_2$ не имеют общих строк. Таким образом, получается новый запрос с путем доступа $П_2$-P. Обозначим через $q_2'$ запрос ($T_L^2$, $T_{con}^2$, $T_{rr}'$, ∅), где $T_{rr}'$ - графическое представление пути доступа $П_2$-P. Обозначим через $T_2'$ таблицу, полученную для запроса $q_2'$ с помощью алгоритма 3.1. Этот случай эквивалентен случаю б), но здесь $T_2'$ имеет меньше строк, чем $T_2$. Поэтому $T_2 --_w T_2'$.

Следовательно, добавляя разветвления к $П_2$ (или удаляя разветвления из $П_2$), мы всегда получаем путь доступа некоторого запроса, который является слабоэквивалентным запросом для $q_2$. Поэтому $q_1 --_w q_2$.

Необходимость. Знаем, что $q_1 --_w q_2$, следовательно, существует "минимальный" запрос $q_m$ такой, что $q_1 --_w q_m$ и $q_2 --_w q_m$. Если через $T_1$, $T_2$ и $T_m$ обозначим соответственно таблицы запросов $q_1$, $q_2$ и $q_m$, то тогда таблица $T_m$ является минимальной слабоэквивалентной таблицей для $T_1$ и $T_2$.

Запросы $q_1$ и $q_2$ заданы в некоторой D-ациклической схеме; следовательно, запросы $q_1$ и $q_2$ тоже являются D-ациклическими запросами. Тогда таблица $T_m$ получается из таблицы $T_1$ (или из $T_2$) с помощью алгоритма 4.2. (Алгоритм 4.2 редуцирует ациклические таблицы до получения минимальной слабоэквивалентной таблицы).

Пусть $T_1 \neq T_m$ и $T_2 \neq T_m$ (случай $T_1 = T_m$ и $T_2 = T_m$ включается в

дальнейших рассуждениях). Тогда $T_m$ имеет меньше строк, чем $T_1$ и $T_2$.

Пусть $T_1 - T_m = \{r_1, r_2, \ldots, r_s\}$. Существует содержащее отображение $h_1 : h_1(T_1) = T_m$. Очевидно $h_1(T_m) = T_m$. Тогда $h_1$ сохраняет все символы, которые участвуют в $T_m$. Пусть $h_1$ отображает строки $\{r_1, r_2, \ldots, r_s\}$ в строки $\{r_1^m, r_2^m, \ldots, r_{k1}^m\}$ из $T_m$, где $s > k_1$. Пусть

$$P_i^1 = \{ r \mid r \in \{r_1, r_2, \ldots, r_s\} \text{ и } h_1(r) = r_i^m \},\ 1 \le i \le k_1.$$

Каждое множество строк $P_i^1$, $1 \le i \le k_1$, редуцируется до получения строки $r_i^m \in T_m$ с помощью алгоритма 4.2. Тогда легко видеть, что $P_i^1$, $1 \le i \le k_1$, являются разветвлениями пути доступа $П_1$; графические представления $P_1^1, P_2^1, \ldots, P_{k1}^1$ имеют не больше чем одну общую строку с графическим представлением запроса $q_m$. (Если $P_1^1, P_2^1, \ldots, P_{k1}^1$ - несвязанные множества строк, то тогда их связанные подмножества строк являются разветвлениями пути доступа $П_1$.)

Пусть $T_2 - T_m = \{r_1', r_2', \ldots, r_p'\}$. Существует содержащее отображение $h_2 : h_2(T_2) = T_m$. Очевидно $h_2(T_m) = T_m$. Тогда $h_2$ сохраняет все символы, которые участвуют в $T_m$. Пусть $h_2$ отображает строки $\{r_1', r_2', \ldots, r_s'\}$ в строки $\{r_1^{m\prime}, r_2^{m\prime}, \ldots, r_{k2}^{m\prime}\}$ из $T_m$, где $p > k_2$. Пусть

$$P_i^2 = \{ r \mid r \in \{r_1', r_2', \ldots, r_p'\} \text{ и } h_2(r) = r_i^{m\prime} \},\ 1 \le i \le k_2.$$

Тогда видно, что $P^2_l$, $1 \le l \le k_2$, являются разветвлениями пути доступа $П_2$, которые имеют не больше чем одну общую строку с графическим представлением запроса $q_m$.

Следовательно, путь доступа запроса $q_m$ получается из путей доступа запросов $q_1$ и $q_2$ с помощью удаления всех разветвлений из запросов $q_1$ и $q_2$. Пусть в пути доступа запроса $q_1$ встречаются разветвления $P^1_1, P^1_2, \ldots, P^1_{k1}$, а в пути доступа запроса $q_2$ - разветвления $P^2_1, P^2_2, \ldots, P^2_{k2}$. Тогда путь доступа $П_1$ получается из $П_2$ с помощью удаления разветвлений $P^2_1, P^2_2, \ldots, P^2_{k2}$ и добавления разветвлений $P^1_1, P^1_2, \ldots, P^1_{k1}$. □

Можно отметить, что алгоритм 4.2 стирает все разветвления запроса (в случае D-ациклического запроса - все разветвления и только их). Доказательство этого факта проводится индуктивно.

## 5.2 Алгоритм проверки эквивалентности запросов в D-ациклических схемах

На основе теоремы 5.1 можно предложить алгоритм проверки эквивалентности запросов в D-ациклических схемах.

Пусть $U$ состоит из $m$ атрибитов, а в схеме базы данных участвуют $n$ реляционные схемы $R_1, R_2, \ldots, R_n$. Тогда таблица доступа, в которой строится графическое представление всех запросов, имеет $m$ столбцов и $n$ строк.

ОПРЕДЕЛЕНИЕ 5.2. Пусть $q=(T_L,T_{con},T_{rr},\varnothing)$ и пусть $T_q$ - таблица запроса q, построеная с помощью алгоритма 3.1. Вводим кодирование K для запроса q - в каждом элементе $a_{ij}$ таблицы доступа записываем цепочку $x^1x^2x^3$, где:

$x^1$ -
- 0, если в $T_q$ в строке $R_i$, в столбце $A_j$ - пробел;
- a, если в $T_q$ в строке $R_i$, в столбце $A_j$ - свободная переменная;
- c, если в $T_q$ в строке $R_i$, в столбце $A_j$ - константа;
- ac, если в $T_q$ в строке $R_i$, в столбце $A_j$ - константа, которая участвует в резюме $T_q$;
- b, в остальных случаях.

$x^2$ -
- $i_p$, если $i_p<i$ и $((R_{ip},A_j),(R_i,A_j))$ является ребром в $T_{rr}$;
- 0, если значение $i_p$ не существует.

$x^3$ -
- $i_t$, если $i_t>i$ и $((R_{it},A_j),(R_i,A_j))$ является ребром в $T_{rr}$;
- 0, если значение $i_t$ не существует. □

Упорядоченная четверка $(T_L,T_{con},T_{rr},\varnothing)$ и таблица $T_q$ определяют однозначно кодирование K. Значения $x^2$ и $x^3$ в цепочке $x^1x^2x^3$ описывают связи элемента $a_{ij}$ с остальными элементами столбца $A_j$ из $T_{rr}$: $x^2$ - с его "предшественниками" в столбце $A_j$,

$x^3$ - с его "наследниками" в столбце $A_j$. Из определений 3.4 и 3.6 видно, что если для некоторого запроса q исполнено $\Sigma-\varnothing$, всегда для него существует кодирование K вышеописанного типа.

ПРИМЕР 5.2. Рассмотрим соответствующие кодирования для запросов $q_1$ и $q_1'$ из примера 5.1 (показаны только ненулевые значения таблиц):

| | ЧН | Ц | ПН | КН |
|---|---|---|---|---|
| ЧАСТЬ | b02 | 200 | | |
| ОБЯЗАННОСТЬ | b1 | | a | |
| ПОСТАВЩИК | | | | |
| ПОСТАВКА | | | | |
| КЛИЕНТ | | | | |

| | ЧН | Ц | ПН | КН |
|---|---|---|---|---|
| ЧАСТЬ | b02 | 200 | | |
| ОБЯЗАННОСТЬ | b1 | | a03 | |
| ПОСТАВЩИК | | | a24 | |
| ПОСТАВКА | | | a30 | b05 |
| КЛИЕНТ | | | | b4 |

□

Рассмотрим лемму, которая уточняет шагов алгоритма проверки эквивалентности.

ЛЕММА 5.1. Пусть $q_1$ и $q_2$ - слабоэквивалентные запросы в некоторой D-ациклической схеме $\mathbf{R}$ и пусть $P_1$ - разветвление запроса $q_1$. ($P_1$ имеет не больше чем одну общую строку с $q_1 \cap q_2$.) Пусть $R_1 \in P_1 \cap (q_1 \cap q_2)$ и пусть $R_2 \in P_1 - (q_1 \cap q_2)$ - реляционная схема из $P_1$, связана с $R_1$. Пусть $\{A_1, A_2, A_3\} \subset R_1 \cap R_2$, где $A_1 \in U$ для $1=1,2,3$ и $A_1 \neq A_j$ для $1 \neq j$. Пусть $R_1$ и $R_2$ связаны в столбце $A_1$ с помощью цепочки

$(R_1.A_1 = R^1_{11}.A_1)$ AND $(R^1_{11}.A_1 = R^1_{12}.A_1)$ AND...AND $(R^1_{1p}.A_1 = R_2.A_1)$;

Пусть $R_1$ и $R_2$ связаны в столбце $A_2$ с помощью цепочки

$(R_1.A_2=R^2_{I1}.A_2)$ AND $(R^2_{I1}.A_2=R^2_{I2}.A_2)$ AND...AND $(R^2_{Is}.A_2=R_2.A_2)$ ;

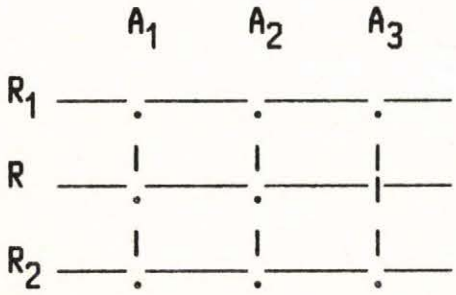
Пусть $R_1$ и $R_2$ связаны в столбце $A_3$ с помощью цепочки

$(R_1.A_3=R^3_{I1}.A_3)$ AND $(R^3_{I1}.A_3=R^3_{I2}.A_3)$ AND...AND $(R^3_{It}.A_3=R_2.A_3)$ .

Тогда

$$\{R^1_{I1},R^1_{I2},\ldots,R^1_{Ip}\}\cap\{R^2_{I1},R^2_{I2},\ldots,R^2_{Is}\}\cap\{R^3_{I1},R^3_{I2},\ldots,R^3_{It}\}=\varnothing .$$

ДОКАЗАТЕЛЬСТВО. Допустим, что существует реляционная схема R такая, что $R=R^1_{Ij}$ и $R=R^2_{Ik}$. Тогда графические представления $q_1\cap q_2$ и $P_1$ можно изобразить следующим образом:




Видно, что $\{A_1,A_2,A_3\}\subset R_1$, $\{A_1,A_2,A_3\}\subset R_2$ и $\{A_1,A_2\}\subset R$.

Можно построить цепочку реляционных схем и их атрибутов $(R_1,A_1,R,A_2,R_2,A_3,R_1)$; из определения 4.16 следует, что схема базы данных R является D-циклической. Это противоречит условию леммы.

Следовательно,

$$\{R^1_{I1},R^1_{I2},\ldots,R^1_{Ip}\}\cap\{R^2_{I1},R^2_{I2},\ldots,R^2_{Is}\}\cap\{R^3_{I1},R^3_{I2},\ldots,R^3_{It}\}=\varnothing .$$

□

Теорема 5.1 и лемма 5.1 определяют возможные связы между элементами разветвлений:

Пусть $q_1 --_w q_2$ и $P_1$ -разветвление запроса $q_1$. Тогда:

а) если в $P_1$ в элементе $a_{IJ}$ встречается некоторая свободная переменная или константа, элемент $a_{IJ}$ должен быть связанным в $P_1$ с элементом $a_{IJ} \in q_1 \cap q_2$, в котором встречается та же самая свободная переменная или константа;

б) если некоторая строка $r \in P_1$ содержит элементы $a_{IJ1}, a_{IJ2}, \ldots, a_{IJs}$, связанные с элементами из $q_1 \cap q_2$, то тогда в $q_1 \cap q_2$ должна участвовать строка $r_m$ содержащая все элементы $a_{IJ1}, a_{IJ2}, \ldots, a_{IJs}$. Если $s \geq 3$, то тогда лемма 5.1 показывает, что $a_{IJ1}, a_{IJ2}, \ldots, a_{IJs}$ должны быть связанными с элементами из $q_1 \cap q_2$ с помощью непересекающихся множеств реляционных схем.

Вводим следующие обозначения: через $a_{IJ}$ и $b_{IJ}$ будем обозначать элементы из $K_1$ и $K_2$, которые встречаются в I-той строке, в J-том столбце. Ниже рассмотрим алгоритм проверки эквивалентности. В нем будем использовать $x^4_{IJ}$ и $y^4_{IJ}$ для поиска несовпадающих разветвлений.

АЛГОРИТМ 5.1. Проверка эквивалентности двух конъюнктивных запросов $q_1$ и $q_2$, в которых каждая реляционная схема участвует не больше одного раза. Запросы $q_1$ и $q_2$ заданы в некоторой D-ациклической схеме реляционной базы данных.

ВХОД: Кодирования $K_1$ и $K_2$ соответственно для запросов $q_1$ и $q_2$.

МЕТОД:

1. Q :- 1;

2. Проверить совпадения множеств $T_L^1$ и $T_L^2$, $T_{con}^1$ и $T_{con}^2$. Если не совпадают, то Q :- 0, конец работы.

Отметить общие для $K_1$ и $K_2$ места - если $a_{IJ} \neq 0$ и $b_{IJ} \neq 0$, то $x_{IJ}^4$ :- 1 и $y_{IJ}^4$ :- 1. В остальных случаях $x_{IJ}^4$ :- 0 и $y_{IJ}^4$ :- 0.

3. Обработать столбцы $K_1$. Для всех элементов $a_{IJ} \neq 0$, где $b_{IJ} \neq 0$, рассмотреть все "предшественники" в столбце J, которые участвуют только в $q_1$ (т.е., у которых $x^4 = 0$); пусть это будут элементы $a_{tJ}$, $t<I$. Тогда если $x_{tJ}^1 \in \{a,c,ac\}$ и $x_{tJ}^1 = x_{IJ}^1$, сделать $x_{tJ}^4$ :- 2.

Рассмотреть все "наследники" в столбце J, участвующие только в $q_1$ (т.е., у которых $x^4 = 0$); пусть это будут элементы $a_{sJ}$, $s>I$. Если $x_{sJ}^1 \in \{a,c,ac\}$ и $x_{sJ}^1 = x_{IJ}^1$, сделать $x_{sJ}^4$ :- 2.

4. Обработать столбцы $K_2$. Для всех элементов $b_{IJ} \neq 0$, где $a_{IJ} \neq 0$, рассмотреть все "предшественники" в столбце J, которые участвуют только в $q_2$ (т.е., у которых $y^4 = 0$); пусть это будут элементы $b_{tJ}$, $t<I$. Тогда если $y_{tJ}^1 \in \{a,c,ac\}$ и $y_{tJ}^1 = y_{IJ}^1$, сделать $y_{tJ}^4$ :- 2.

Рассмотреть все "наследники" в столбце J, участвующие только в $q_2$ (т.е., у которых $y^4_{sJ}=0$); пусть это будут элементы $b_{sJ}$, $s>1$. Если $y^1_{sJ}\in\{a,c,ac\}$ и $y^1_{sJ}=y^1_{1J}$, сделать $y^4_{sJ}$ :- 2.

5. Рассмотреть все элементы $a_{1J}\in K_1$, $a_{1J}\neq 0$. Если $x^1_{1J}\in\{a,c,ac\}$, то должно быть исполнено $x^4_{1J}=1$ или $x^4_{1J}=2$. Иначе Q :- 0; конец работы.

6. Рассмотреть все элементы $b_{1J}\in K_2$, $b_{1J}\neq 0$. Если $y^1_{1J}\in\{a,c,ac\}$, то должно быть исполнено $y^4_{1J}=1$ или $y^4_{1J}=2$. Иначе Q :- 0; конец работы.

7. Рассмотреть строки $K_1$: $r_1,r_2,\ldots,r_n$.

7.1. Если в строке $r_1$ для всех элементов $a_{1J}$, $1\leq J\leq m$, исполнено $x^4_{1J}=0$ или $x^4_{1J}=1$, перейти к рассмотрению следующей строки $K_1$.

7.2. Если в строке $r_1$ для $1\leq J\leq m$ встречаются только значения $x^4_{1J}=0$ или $x^4_{1J}=2$, сделать следующее:

а) если существует только одно значение $J_0$: $x^4_{1J_0}=2$, то перейти к рассмотрению следующей строки $K_1$.

б) если существуют значения $J_1, J_2, \ldots, J_p$ такие, что $x^4_{IJ1}=x^4_{IJ2}=\ldots=x^4_{IJp}=2$, построить множества $N_t$ для $1\le t\le p$:

$N_t = \{ l \mid a_{lJt}\in K_1$ для $1\le l\le n$, $x^4_{lJt}=1$ и $a_{lJt}$ связан с $a_{IJt}$ в столбце $J_t \}$.

(Как показывает лемма 5.1, множества $N_t$, $1\le t\le p$, имеют не больше чем один общий элемент.)

Если $N_1\cap N_2\cap\ldots\cap N_p=\varnothing$, то $q_1\neq q_2$, Q:-0, конец работы.

Если $N_1\cap N_2\cap\ldots\cap N_p\neq\varnothing$, перейти к рассмотрению следующей строки $K_1$.

7.3. Если в строке $r_I$ для $1\le J\le m$ встречаются значения $x^4_{IJ}=0$, $x^4_{IJ}=1$ и $x^4_{IJ}=2$, сделать следующее: построить множества $N_t$ для $1\le t\le p$, где $x^4_{It}\neq 0$ для $1\le t\le p$:

$N_t = \{l \mid a_{lJt}\in K_1$ для $1\le l\le n$, $x^4_{lJt}=1$ и $a_{lJt}$ связан с $a_{IJt}$ в столбце $J_t\}$.

Если $N_1\cap N_2\cap\ldots\cap N_p=\varnothing$, то $q_1\neq q_2$, Q :- 0, конец работы.

Если $N_1\cap N_2\cap\ldots\cap N_p\neq\varnothing$, перейти к рассмотрению следующей строки $K_1$.

8. Рассмотреть строки $K_2$: $r_1', r_2', \ldots, r_n'$.

8.1. Если в строке $r_I'$ для всех элементов $b_{IJ}$, $1\le J\le m$, исполнено $y^4_{IJ}=0$ или $y^4_{IJ}=1$, перейти к рассмотрению следующей

строки $K_2$.

8.2. Если в строке $r_1$, для $1 \le J \le m$ встречаются только значения $y^4_{1J}=0$ или $y^4_{1J}=2$, сделать следующее:

а) если существует только одно значение $J_0$: $y^4_{1J0}=2$, то перейти к рассмотрению следующей строки $K_2$.

б) если существуют значения $J_1, J_2, \ldots, J_p$ такие, что $y^4_{1J1}=y^4_{1J2}=\ldots=y^4_{1Jp}=2$, построить множества $N_t$ для $1 \le t \le p$:

$N_t = \{ l \mid b_{lJt} \in K_1$ для $1 \le l \le n$, $y^4_{lJt}-1$ и $b_{lJt}$ связан с $b_{1Jt}$ в столбце $J_t \}$.

(Как показывает лемма 5.1, множества $N_t$, $1 \le t \le p$, имеют не больше чем один общий элемент.)

Если $N_1 \cap N_2 \cap \ldots \cap N_p = \varnothing$, то $q_1 \neq q_2$, Q :- 0, конец работы.

Если $N_1 \cap N_2 \cap \ldots \cap N_p \neq \varnothing$, перейти к рассмотрению следующей строки $K_2$.

8.3. Если в строке $r_1$, для $1 \le J \le m$ встречаются значения $y^4_{1J}=0$, $y^4_{1J}=1$ и $y^4_{1J}=2$, сделать следующее: построить множества $N_t$ для $1 \le t \le p$, где $y^4_{1t} \neq 0$ для $1 \le t \le p$:

$N_t = \{ l \mid b_{lJt} \in K_1$ для $1 \le l \le n$, $y^4_{lJt}-1$ и $b_{lJt}$ связан с $b_{1Jt}$ в столбце $J_t \}$.

Если $N_1 \cap N_2 \cap \ldots \cap N_p = \varnothing$, то $q_1 \neq q_2$, Q :- 0, конец работы.

Если $N_1 \cap N_2 \cap \ldots \cap N_p \neq \varnothing$, перейти к рассмотрению следующей строки $K_1$.

ВЫХОД: Если Q=0, то $q_1 \neq q_2$. Если Q=1, то $q_1 --_ш q_2$. □

Отметим, что длина входа алгоритма 5.1 - 2nm. На втором шагу алгоритм 5.1 обработывает 2nm элементов. На третьем шагу для каждого столбца рассматриваются цепочки связанных в столбце элементов. Если в столбце существуют разные цепочки, то из определения 3.5 видно, что они связывают разные вершины графического преставления и поэтому являются непересекающимися цепочками вершин. Для каждого столбца рассматриваются не больше чем 2n элементов на третьем шагу и поэтому в общем обработываются 2nm элементов (как и на четвертом шагу). На пятом и шестом рассматриваются 2nm элементов. На седмом шагу для каждой строки нужно обработать nm элементов, как и на восьмом шагу; поэтому на седьмом и восьмом шагу алгоритм 5.1 рассматривает $2n^2m$ элементов.

# Г Л А В А  Ш Е С Т А Я

# ИНТЕРФЕЙСЫ ВЫСОКОГО УРОВНЯ К РЕЛЯЦИОННЫМ БАЗАМ ДАННЫХ

В последнее время большую популярность приобрели попытки построения интерфейсов высокого уровня. Например в исследованиях по искусственному интеллекту рассматриваются проблемы построения интерфейса на естественном языке к реляционным базам данных. Также при проектировании больших систем (и новых поколений компьютеров) реляционные системы баз данных обычно рассматриваются в качестве обязательного элемента, обеспечивающего сохранение и обработку основных объемов фактов.

Автор с сожалением констатирует, что в обоих вышеупомянутых случаях не уделяется необходимое внимание проблеме логической навигации.

Например при построении интерфейса на естественном языке к реляционной базе данных полагается, что пользователь задает свой запрос к базе данных на естественном языке. При этом надо отметить, что пользователь описывает на естественном языке обычно множества $T_L$, $T_{con}$ и $\Sigma$ запроса к реляционной базе данных и не описывает путь доступа $T_{rr}$. Как показывают исследования, пользователь обычно подразумевает семантические связи между отдельными элементами запросов. Следовательно, при построении интерфейса на естественном языке основной проблемой является реализация алгоритмов автоматического осуществления логической

навигации. Разумеется, интерфейс на естественном языке осуществляется всегда в диалоговом режиме и всегда можно предоставить пользователю возможность выбора между несколькими вероятными путями доступа. Этот подход не может быть применен, однако, для пользователей, которые не знакомы в достаточной степени с организацией данных.

Нужно отметить, что разработываемые до сих пор интерфейсы на естественном языке являются преимущественно разработками исследовательского характера, которые не обладают большим прикладным значением. В доступных источниках обычно рассматриваются примерные запросы, для которых нужно извлечь информацию только из одного отношения базы данных. Конечно, в этом случае не нужно осуществлять логической навигации.

При проектировании больших систем (и новых поколений компьютеров) реляционные базы данных рассматриваются в качестве основного элемента построения, обеспечивающего доступ к большим объемам фактов. Естественно, в этом случае тоже нужно создать процедуры автоматической реализации логической навигации.

В последнее время предлагается решать проблему логической навигации, используя универсальной реляции. Описание таких интерфейсов к реляционной базе данных дано например в [KKFGU84] и [АнЖ87].

Автор настоящей диссертации работал над проблемами построения интерфейса на естественном языке к реляционной базе данных. В [An981], [An983] и [PAn83] рассмотрены проблемы, связанные с построением лингвистических процессоров для доступа на болгарском языке к реляционным базам данных. В [PAP84] рассмотрены проблемы, связанные с автоматическим анализированием болгарского языка. В [PAP84] и [StAn86] описана система ML-1, разработана в Лаборатории математической лингвистики Института Математики Болгарской Академии Наук.

Система ML-1 является диалоговой системой информационно-справочного типа; доступ к ней можно осуществлять с помощью реляционного языка QUEL. Система ML-1 используется в качестве основы для построения лингвистического процессора для доступа к реляционной базе данных на болгарском языке. В [StAn86] описаны возможности реализации процедур, поддерживающих некоторую целостность базы данных в системе ML-1. Эти процедуры обеспечивают поддержку универсальной реляции в базе данных и, следовательно, эти процедуры обеспечивают слабую эквивалентность запросов к системе ML-1.

На основании полученных в настоящей диссертации результатов можно предложить более конкретный подход при построении интерфейсов высокого уровня к реляционным базам данных.

Самое эффективное применение полученных результатов можно осуществить в D-ациклических схемах реляционных баз данных. Как видно из теоремы 4.5, в D-ациклических схемах можно задавать только D-ациклические запросы (которые очевидно являются ациклическими запросами). В D-ациклических схемах очень легко можно реализовать алгоритмы проверки эквивалентности для конъюнктивных запросов. Как показывают теорема 5.1 и алгоритм 5.1, в D-ациклических схемах существует алгоритм проверки эквивалентности конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза. Этот алгоритм работает в полиномиальном периоде времени.

Следовательно, если в процессе проектирования схема реляционной базы данных получится D-ациклической, в соответствующей системе управления базы данных можно сгенерировать эффективные алгоритмы для оптимизации и проверки эквивалентности запросов. Алгоритм 4.2 может быть использован для оптимизации конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза. Алгоритм 5.1

может быть использован для проверки эквивалентности двух заданных конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза.

Если в процессе проектирования схема реляционной базы данных не получится D-ациклической, то можно использовать следующий алгоритм для оптимизации конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза:

АЛГОРИТМ 6.1. Оптимизация относительно слабой эквивалентности конъюнктивных запросов, в которых каждая реляционная схема участвует не больше одного раза. Оптимизируются запросы, заданные в D-циклических реляционных схемах.

ВХОД. Таблица конъюнктивного запроса, построена по алгоритму 3.1.

МЕТОД.

1. Запустить алгоритм 4.2 над входной таблицей.

2. Если таблица является ациклической, тогда полученная таблица GR(T) является минимальной слабоэквивалентной таблицей. Конец работы.

3. Запустить алгоритм 4.3 над таблицей GR(T) до получения минимальной слабоэквивалентной таблицы.

ВЫХОД. Минимальная таблица, слабоэквивалентна данной таблицей.

Алгоритмы 4.2 и 4.3 всегда заканчивают работу в полиномиальном периоде времени; поэтому алгоритм 6.1 тоже всегда заканчивает работу в полиномиальном периоде времени.

В диссертации рассматривается проблема эквивалентности и оптимизации запросов в реляционной базе данных.

В основе настоящей работы лежат следующие идеи:

1. Рассмотреть все запросы к реляционным базам данных как класс объектов действительности, который может быть описан с помощью различных способов - т.е., с помощью различных реляционных языков запросов. Запросы к реляционным базам данных существуют независимо от способов их описания.

2. Языки описания запросов к реляционным базам данных могут быть исследованы путем сравнения. В работе показано, что оптимизацию конъюнктивных запросов в некоторых случаях можно начать на синтаксическом уровне, т.е. различные появления некоторой реляционной схемы в конъюнктивном запросе можно рассматривать как одно появление.

Полученные в работе результаты можно интерпретировать с точки зрения искусственного интеллекта. Как известно [BMS84], в последнее время появилось много работ, в которых предлагается описывать проблемную область базы данных средствами представления знаний искусственного интеллекта. Например, в [BMS84] в качестве семантической модели проблемной области использованы семантические сети. Таким образом представление запроса в виде гиперграфа позволяет рассматривать каждый запрос в виде семантической сети, заданной на семантической сети

проблемной области. Минимизацию конъюнктивных запросов можно рассматривать как процесс редуцирования семантической сети запроса, причем видно, что сеть запроса состоит из:

а) "существенных" объектов - с их устранением запрос меняет смысл;

б) "несущественных" объектов - их устранение не приводит к перемене смысла запроса.

По мнению автора, в настоящей работе получены два очень важных оригинальных результата:

а) введен алгоритм оптимизации запросов к реляционной базе данных, который отличается от известных до сих пор алгоритмов и основывается на операциях "поглощение" и "удаление изолированных символов".

б) введена интерпретация запроса в виде гиперграфа, которая позволяет поставить вопрос: "Какие запросы могут быть сформулированы к разным видам реляционных схем баз данных? Интерпретация запроса в виде гиперграфа позволяет отделить класс запросов и класс схем, для которых существует алгоритм проверки эквивалентности в полиномиальном периоде времени, при чем некоторые синтаксические конструкции запроса могут быть удалены еще на синтаксическом уровне. В доказательствах существенно используется вышеупомянутый алгоритм оптимизации.

В [Sag81] сформулирован для исследования следующий вопрос:

"Оптимизировать запросы относительно более дешевого критерия, который зависит тоже от операндов (а не только от операторов в реляционном выражении). Найти подходящий критерий этого вида. Этот критерий будет использован в рассмотрении специфических применений".

Можно сказать, что настоящая диссертация дает ответ на этот

вопрос, особено с введением D-ациклических схем и запросов.

Представление запроса в виде гиперграфа позволяет нам выявить новые проблемы для исследования:

1. Найти необходимые и достаточные условия эквивалентности запросов в других видах схем, в которых могут участвовать и циклические компоненты. (Достаточное условие этого типа дано в [An986] и в параграфе 5.3). Очевидно, в случае схем с циклическими компонентами, разветвления путей доступа могут содержать и циклические компоненты.

2. Исследовать вопрос оптимизации при наличии ограничений. Проблема проверки эквивалентности запросов может получить новое разрешение при наличии, например, функциональных зависимостей и зависимостей других видов. В [LMG83] и в [Lav85] сделан шаг в этом направлении.

3. Проблема проверки эквивалентности и оптимизации традиционно решается для запросов $q=(T_L, T_{con}, T_{rr}, \Sigma)$, где $\Sigma=\varnothing$. Другими словами, рассматривается прежде всего естественное соединение $\bowtie$. Найти эффективные алгоритмы оптимизации запросов, для которых $\Sigma\neq\varnothing$. В [Ull82] сделан шаг в этом направлении, но не исследованы возможности построения алгоритмов для оптимизации в полиномиальном периоде времени.

# Л И Т Е Р А Т У Р А

[AC75] Astrahan, M.M., and Chamberlin, D.D. Implementation of a Structured English Query Language. CACM Vol. 18, No. 10, October 1975, pp. 580-587.

[ASU79] Aho, A., Sagiv, Y. and Ullman, J. Equivalence among Relational Expressions. SIAM Journal of Computing, Vol.8, No.2, May 1979, pp. 218-246.

[ASU79a] Aho,A., Sagiv, Y. and Ullman, J. Efficient Optimization of a Class of Relational Expressions. ACM Transactions on Data Base Systems, Vol.4, No.4, December 1979, pp. 435-454.

[Ang81] Angelova, G. The Use of Natural Language as a Query Language in a Relational Data Base. Proceedings of the Fourth International Seminar on Data Base Management Systems, Schwerin, GDR, December 1981, pp. 226-238.

[Ang83] Angelova, G. About Some Problems of the Automatical Generation of Queries to a Relational Data Base. Proceedings of the Sixth International Seminar on Data Base Management Systems, Matrafured, Hungary, October 1983, pp. 27-36.

[Ang86] Angelova, G. On the Equivalence of Conjunctive Queries to a Relational Data Base. Proceedings of the Fourth Hungarian Computer Science Conference, M. Arato, I. Katai, L. Varga (eds.), Budapest, Akademiai Kiado, 1986, pp. 361 - 371.

[Ang87] Angelova, G. Optimization of a Class of Conjunctive Queries to a Relational Data Base (to appear).

[Ang87a] Angelova, G. Types Equivalent Queries for a Class of Conjunctive Queries to a Relational Data Base (to appear).

[AnZ85] Angelov, Zh. Towards a Universal Relation View. Proceedings of the Eigth International Seminar on Data Base Management Systems, Piestany, CSSR, September 1985, pp. 9-17.

[ABe81] Arato, M. and Benczur, A. Dynamic Placement of Records and the Classical Occupancy Problem, Comp. and Math. with Appls., No. 7, 1981, pp. 173-185.

[ABe82] Arato, M. and Benczur, A. A General Treatement of Rearrangement Problems in a Linear Storage. Performance Evaluation, No. 2, 1982, pp. 108-117.

[BCKH75] Boyce, R.F., Chamberlin, D.D., King, W.F., and Hammer, M.M. Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage. CACM Vol. 18, No. 11, November 1975, pp. 621-628.

[BFMY83] Beeri, C., Fagin, R., Maier, D. and Yannakakis, M. On the Desirability of Acyclic Database Schemes. Journal of the ACM, Vol. 30, No.3, July 1983, pp. 479-513.

[BFMMUY81] Beeri, C., Fagin, R., Maier, D., Mendelzon, A., Ullman, J. and Yannakakis, M. Properties of Acyclic Database Schemes. Proceedings of the Thirteenth Annual ACM Symposium

on the Theory of Computing (1981), pp. 355-362.

[Ben83] Benczur, A. Problems in Modeling of Data Base Performance. Perspectives in Developing Computer Technics, Moscow, URSS, 1983, pp. 85-98.

[BeS84] Benczur, A. and Stahl, J. On Updating a Large-scale Data System. Alkalmazott Matematikai Lapok, No.10, 1984, pp.1-13 (in Hungarian).

[BDe79] Bekessy, A. and Demetrovich, J. Contribution to the Theory of Data Base Relations. Discrete Mathematics, North-Holland, No. 27, 1979, pp. 1-10.

[BMS84] Brodie, M., Mylopoulos, J. and Schmidt, J. (Eds.) On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages. Springer-Verlag, 1984.

[CLV85] Chung Le Viet. Translation and Compatibility of SQL and QUEL Queries. Journal of Information Processing, Vol. 8, No. 1, 1985, pp. 1-15.

[CeG85] Ceri, S. and Gottlob, G. Translating SQL into Relational Algebra: Optimization, Semantics and Equivalence of SQL Queries. IEEE Transactions of Software Engineering, Vol. 11, No. 4, 1985, pp. 324-345.

[Cod70] Codd, E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

[Cod72] Codd, E. F. Further Normalization of the Data Base Relational Model. In Data Base Systems, Courant Computer Science Symposia, Vol. 6, Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 33-64.

[Cod71] Codd, E. F. A Data Base Sublanguage Founded on the Relational Calculus. ACM SIGFIDET Workshop on Data Description, Access and Control, November 1971, pp. 35-61.

[Cod72a] Codd, E. F. Relational Completeness of Data Base Sublanguages. In Data Base Systems, Courant Computer Science Symposia, Vol. 6, Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 65-98.

[Dem78] Demetrovich, J. On the Number of Candidate Keys. Information Processing Letters, Vol. 7, No. 6, 1978, pp. 266-269.

[Dem79] Demetrovich, J. On the Equivalence of Candidate Keys with Sperner Systems. Acta Cybernetica, Szeged. Vol. 4, No. 3, 1979, pp. 247-252.

[DeG81] Demetrovich, J. and Gyepesi, G. On the Functional Dependency and Some Generalizations of It. Acta Cybernetica, Szeged, Vol. 5, No.3, 1981, pp. 295-305.

[DeG81a] Demetrovich, J. and Gyepesi, G. Axiomatization of General Functional Dependencies in Relational Data Bases. Kibernetika, Kiev, No. 2, 1981, pp. 42-48.

[Dem81] Demetrovich, J. Mathematical Analysis of the Relational Data Model. Proceedings of the Third Hungarian Computer Science Conference, Budapest, 1981, pp. 65-73.

[FMU82] Fagin, R., Mendelzon, A. and Ullman, J. A Symplified Universal Relation Assumption and its Properties. ACM Transactions on Data Base Systems, Vol.7, No.3, September 1982.

[Fag83] Fagin, R. Degrees of Acyclicity for Hypergraphs and Relational Data Base Schemes. Journal of the ACM, Vol. 30, No.3, July 1983, pp. 514-550.

[Fag83a] Fagin, R. Acyclic Data Base Schemes of Various Degrees - a Painless Introduction. Lecture Notes in Computer Science, G. Ausiello and M. Protassi (eds.), Vol.159, 1983, pp. 65-89.

[FVa84] Fagin, R. and Vardi, M. The Theory of Data Dependencies - a Survey. IBM Research Report No. 4321, 1984.

[GoS82a] Goodman, N. and Shmueli, O. Tree Queries - a Simple Class of Relational Queries. ACM Transactions on Data Base Systems, Vol.7, No.4, December 1982, pp. 653-677.

[Gra80] Graham, M. On the Universal Relation. In: A Panache of DBMS Ideas III, D. Tsichritzis (ed.), Technical Report CSRG-111, April, 1980, University of Toronto.

[GrM82] Graham, M. and Mendelzon, A. Strong Equivalence of Relational Expressions Under Dependencies. Information Processing Letters, Vol. 14, No. 2, 1982, pp. 57-62.

[IL183] Imielinski,T. and Lipski, W. On the Undecidability of the Equivalence Problems for Relational Expressions. In Advances in Database Theory, Vol. II, Galaire, H., Mainker, J. and Nicolas, J. M. (eds.), Academic Press, 1983.

[KKFGU84] Korth, N., Kuper, G., Feigenbaum, J., Val Gelder, A. and Ullman, J. System/U - a Database System Based on the Universal Relation Assumption. ACM Transactions on Database Systems, Vol. 9, No.3, September 1984, pp. 331-347.

[LMG83] Laver, K., Mendelzon, A. and Graham, M. Functional Dependencies on Cyclic Database Schemes. Proceedings of the ACM SIGMOD'83 Annual Meeting, San Jose, May 23-26, 1983, pp. 79-91.

[Lav85] Laver, K. Semantic and Syntactic Properties of Universal Relation Scheme Data Bases. Ph.d. thesis, Technical Report CSRI-163, April 1985, University of Toronto.

[Mai83] Maier, D. The Theory of Relational Databases. Computer Science Press, Rockville, Maryland, 1983.

[MUV84] Maier, D., Ullman, J. and Vardi, M. On the Foundations of the Universal Relation Model. ACM Transactions on Data Base Systems, Vol.9, No.2, June 1984, pp. 283-308.

[MaUl84] Maier, D. and Ullman, J. Connections in Acyclic Hypergraphs. Theoretical Computer Science 32 (1984), North-Holland, pp. 185-199.

[OtHo85] Ott, N. and Horlander, K. Removing Redundant Join Operations in Queries Involving Views. Information Systems, Vol. 10, No. 3, 1985, pp. 279-288.

[PAn83] Pavlov, R. and Angelova, G. Linguistic Processors. MTA SZTAKI Tanulmanyok No.147, 1983, pp. 193-200.

[PAP84] Pavlov, R., Angelova, G. and Paskaleva, E. On Experimental Linguistic Processors for Man-Computer Dialogue in Bulgarian. Proceedings of the International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'84), Varna, Bulgaria, 17-20 September, 1984.

[Ris79] Rissanen, J. The Theory of Relations for Databases - a Tutorial Survey. Lecture Notes in Computer Science, Vol. 64, 1979, pp. 537-551.

[Sag81] Sagiv, Y. Optimization of Queries in Relational Databases. UMI Research Press, Ann Arbor, Michigan, 1981.

[Sag81a] Sagiv, Y. Can We Use the Universal Instance Assumption without Using Nulls? Proceedings ACM SIGMOD International Symposium of Management of Data, 1981, pp. 108-120.

[SaYa80] Sagiv, Y. and Yannakakis, M. Equivalence among Relational Expressions with the Union and Difference Operators. Journal of the ACM, Vol. 27, No. 4, October 1980, pp. 633-655.

[SWKH76] Stonebraker, M., Wong, E., Kreps, P. and Held, G. The

Design and Implementation of INGRES. ACM Transactions on Data Base Systems Vol.1, No.3, September 1976, pp. 189-222.

[StAn86] Stranjev, P. and Angelova, G. A Project for Data Integrity Support in the System ML-1. MTA SZTAKI Tanulmanyok No. 183, 1986, pp. 119-125.

[TuL85] Turner, R. and Lowden, B.G.T. Introduction to the Formal Specification of Relational Query Languages. The Computer Journal, Vol. 28, No. 2, 1985.

[Ull82] Ullman, J. Principles of Database Systems. Computer Science Press, 1982.

[WoM77] Wong, H. K. T., and Mylopoulos, J. Two Views of Data Semantics: Data Models in Artificial Intelligence and Database Management. INFOR, Vol. 15, No.3, 1977.

[WoY76] Wong, E. and Youssefi, K. Decomposition - a Strategy for Query Processing. ACM Transactions on Data Base Systems, Vol. 1, No.3, September 1976, pp. 223-241.

[Yan81] Yannakakis, M. Algorithms for Acyclic Database Schemes. Proceedings 1981 Very Large Data Bases Conference, pp. 82-94.

[Анг78] Ангелова, Г. Средства за контрол на данни - входно-изходна подсистема и елементи за контрол в бази от данни. Дипломна работа, Софийски Университет "Кл. Охридски", 1978.

[АнД81] Ангелова, Г. и Думков, Пл. Алгоритъм за търсене в релационни бази от данни. София, Системи и управление, No.

3, 1981, стр. 26-34.

[Анг86] Ангелова, Г. Таблицы конъюнктивных запросов и их применение. MTA SZTAKI Kozlemenyek, Budapest, 34(1986), стр. 5-35.

[Анг87] Ангелова, Г. Циклические и ациклические схемы реляционных баз данных. MTA SZTAKI Kozlemenyek, Budapest, 36(1987), стр. 5-30.

[Анж87] Ангелов, Ж. Относно универсалната релация като потребителска представа. Кандидатска дисертация, Институт по математика на БАН, 1987.

[ГДж82] Гэри, М. и Джонсон, Д. Вычислительные машины и труднорешаемые задачи. Москва, "Мир", 1982.

[ДПД84] Денев, И., Павлов, Р. и Деметрович, Я. Дискретна математика. София, "Наука и изкуство", 1984.

[Дри82] Дрибас, В. П. Реляционные модели баз данных. Минск, Издательство БГУ им. В.И.Ленина, 1982.

[Кри78] Кристофидес, Н. Теория графов. Москва, "Мир", 1978.

[Рач78] Рачко, Р. Информационная модель роста дерева. Построение модели, Журнал общей биологии. Том XXXIX (1978), Москва, стр. 563-571.

[Улл80] Ульман, Дж. Принципы реляционных баз данных. Москва, "Мир", 1982.

## A TANULMÁNYOK SOROZATBAN 1987-BEN MEGJELENTEK

195/1987 Telegdi László: Bináris változók strukturájának vizsgálata

196/1987 Rónyai Lajos: Algebrai algoritmusok

197/1987 Hernádi Ágnes - Bodó Zoltán - Knuth Előd: A tudásábrázolás technikái és gépi eszközei

198/1987 Miguel Fonfria Atan: A data base management system developed for the Cuban minicomputer CID 300/10

199/1987 Bach Iván - Farkas Ernő - Naszódi Mátyás: A magyar nyelv elemzése számitógéppel

200/1987 Publikációk'86
Szerkesztette: Petróczy Judit

201/1987 Eszenszki József - Hévizi László - dr. Kas Iván - dr. Laufer Judit - Palotási András - Szőnyi Tamás - Dr. Vörös Károly: Tanulmányok a számitástechnika nyomdaipari alkalmazásához

202/1987 PROBLEMS OF COMPUTER SCIENCE
Proceedings of the joint workshop of Computer and Automation Institute of HAS and Computing Centre of Armenian Academy of Sciences held in Budapest, September, 1987.
Edited by: G.B. Marandzjan, B. Uhrin

## 1988-BAN EDDIG MEGJELENTEK:

203/1988 KNVVT EG-25
Problems and tools of the integration of information systems. Proceedings. 1987.
Edited by: Rumjana Kirkova, Tibor Remzső, Ferenc Urbánszki

204/1988 Csetverikov Dimitrij: Digitális texturavizsgálat néhány uj módszere

205/1988 Hernádi Ágnes: Uj eszközök a fogalmi modellezésben

206/1988 The second Hungarian workshop on image analysis
Edited by: Csetverikov Dimitrij, Álló géza

207/1988 Suzanne Márkusz - Gábor Márkus:
Logic Puzzless and Logic Programming I

208/1988 Proceedings of the 5th International Meeting of Young Computer Scientists /IMICS'88/
Edited by: E. Csuhaj-Varju, J. Demetrovics, J. Kelemen
/Smolenice Castle, Czechoslovakia, November 14-18, 1988/