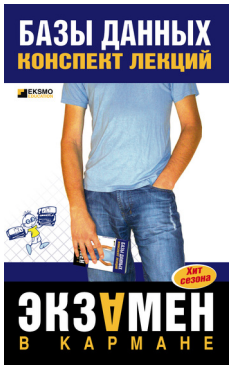

БАЗЫ ДАННЫХ
КОНСПЕКТ ЛЕКЦИЙ
EKSMO
Хит сезона
ЭКЗАМЕН
В КАРМАНЕ

Коллектив авторов

Базы данных: конспект лекций

Базы данных. Конспект лекций: Эксмо; Москва; 2007
ISBN 978-5-699-23778-4

Аннотация

Конспект лекций соответствует требованиям Государственного образовательного стандарта высшего профессионального образования РФ и предназначен для освоения студентами вузов специальной дисциплины «Базы данных».

Лаконичное и четкое изложение материала, продуманный отбор необходимых тем позволяют быстро и качественно подготовиться к семинарам, зачетам и экзаменам по данному предмету.

Содержание

Базы данных: конспект лекций

Лекция № 1. Введение

1. Системы управления базами данных

Системы управления базами данных (СУБД) – это специализированные программные продукты, позволяющие:

1) постоянно хранить сколь угодно большие (но не бесконечные) объемы данных;

2) извлекать и изменять эти хранящиеся данные в том или ином аспекте, используя при этом так называемые запросы;

3) создавать новые базы данных, т. е. описывать логические структуры данных и задавать их структуру, т. е. предоставляют интерфейс программирования;

4) обращаться к хранящимся данным со стороны нескольких пользователей одновременно (т. е. предоставляют доступ к механизму управления транзакциями).

Соответственно, **базы данных** – это наборы данных, находящиеся под контролем систем управления.

Сейчас системы управления базами данных являются наиболее сложными программными продуктами на рынке и составляют его основу. В дальнейшем предполагается вести разработки по сочетанию обычных систем управления базами данных с объектно-ориентированным программированием (ООП) и интернет-технологиями.

Изначально СУБД были основаны на **иерархических** и **сетевых моделях данных**, т. е. позволяли работать только с древовидными и графовыми структурами. В процессе развития в 1970 г. появились системы управления базами данных, предложенные Коддом (Codd), основанные на **реляционной модели данных**.

2. Реляционные базы данных

Термин «реляционный» произошел от английского слова «relation» – «отношение».

В самом общем математическом смысле (как можно помнить из классического курса алгебры множеств) **отношение** – это множество

$$R = \{(x_1, \ldots, x_n) \mid x_1 \# A_1, \ldots, x_n \# A_n\},$$

где $A_1, \ldots, A_n$ — множества, образующие декартово произведение. Таким образом, **отношение R** — это подмножество декартова произведения множеств: $A_1 \times \ldots \times A_n$:

$$R \# A_1 \times \ldots \times A_n.$$

Например, рассмотрим бинарные отношения строгого порядка «больше» и «меньше» на множестве упорядоченных пар чисел $A_1 = A2 = \{3, 4, 5\}$:

$$R_> = \{(3, 4), (4, 5), (3, 5)\} \# A_1 \times A_2;$$
$$R_< = \{(5, 4), (4, 3), (5, 3)\} \# A_1 \times A_2.$$

Эти же отношения можно представить в виде таблиц.

Отношение «больше» $R_>$:

5	4
4	3
5	3

Отношение «меньше» $R_<$:

3	4
4	5
3	5

Таким образом, мы видим, что в реляционных базах данных самые различные данные организовываются в виде отношений и могут быть представлены в форме таблиц.

Нужно заметить, что эти два рассмотренных нами отношения $R_>$ и $R_<$ не эквивалентны между собой, другими словами, таблицы, соответствующие этим отношениям, не равны друг другу.

Итак, формы представления данных в реляционных БД могут быть разными. В чем проявляется эта возможность различного представления в нашем случае? Отношения $R_>$ и $R_<$ – это множества, а множество – структура неупорядоченная, значит, в таблицах, соответ-

ствующих этим отношениям, строки можно менять между собой местами. Но в то же время элементы этих множеств – это упорядоченные наборы, в нашем случае – упорядоченные пары чисел 3, 4, 5, значит, столбцы менять местами нельзя. Таким образом, мы показали, что представление отношения (в математическом смысле) в виде таблицы с произвольным порядком строк и фиксированным числом столбцов является приемлемой, правильной формой представления отношений.

Но если рассматривать отношения $R_>$ и $R_<$ с точки зрения заложенной в них информации, то понятно, что они эквивалентны. Поэтому в реляционных базах данных понятие «отношение» имеет несколько другой смысл, нежели отношение в общей математике. А именно оно не связано с упорядоченностью по столбцам в табличной форме представления. Вместо этого вводятся так называемые схемы отношений «строка – заголовок столбцов», т. е. каждому столбцу дается заголовок, после чего их можно беспрепятственно менять местами.

Вот как будут выглядеть наши отношения $R_>$ и $R_<$ в реляционной базе данных.

Отношение строгого порядка (вместо отношения $R_>$):

Оценка большая	Оценка меньшая
5	4
4	3
5	3

Отношение строгого порядка (вместо отношения $R_<$):

Оценка большая	Оценка меньшая
3	4
4	5
3	5

Обе таблицы-отношения получают новое (в данном случае одинаковое, так как введением дополнительных заголовков мы стерли различия между отношениями $R_>$ и $R_<$) название.

Итак, мы видим, что при помощи такого несложного приема, как дополнение таблиц необходимыми заголовками, мы приходим к тому, что отношения $R_>$ и $R_<$ становятся эквивалентными друг другу.

Таким образом, делаем вывод, что понятие «отношение» в общем математическом и в реляционном смысле совпадают не полностью, не являются тождественными.

В настоящее время реляционные системы управления базами данных составляют основу рынка информационных технологий. Дальнейшие исследования ведутся в направлении сочетания той или иной степени реляционной модели.

Лекция № 2. Отсутствующие данные

В системах управления базами данных для определения отсутствующих данных описаны два вида значений: пустые (или Empty-значения) и неопределенные (или Null-значения).

В некоторой (преимущественно коммерческой) литературе на Null-значения иногда ссылаются как на пустые или нулевые значения, однако это неверно. Смысл пустого и неопределенного значения принципиально различается, поэтому необходимо внимательно следить за контекстом употребления того или иного термина.

1. Пустые значения (Empty-значения)

Пустое значение – это просто одно из множества возможных значений какого-то вполне определенного типа данных.

Перечислим наиболее «естественные», непосредственные **пустые значения** (т. е. пустые значения, которые мы могли бы выделить самостоятельно, не имея никакой дополнительной информации):

1) 0 (нуль) – нулевое значение является пустым для числовых типов данных;

2) false (неверно) – является пустым значением для логического типа данных;

3) B’’ – пустая строка бит для строк переменной длины;

4) “” – пустая строка для строк символов переменной длины.

В приведенных выше случаях определить, пустое значение или нет, можно путем сравнивания имеющегося значения с константой пустого значения, определенной для каждого типа данных. Но системы управления базами данных в силу реализованных в них схем долговременного хранения данных могут работать только со строками постоянной длины. Из-за этого пустой строкой бит можно назвать строку двоичных нулей. Или строку, состоящую из пробелов или каких-либо других управляющих символов, – пустой строкой символов.

Вот несколько примеров пустых строк постоянной длины:

1) B’0’;
2) B’000’;
3) ‘ ‘.

Как же в этих случаях определить, является ли строка пустой?

В системах управления базами данных для проверки на пустоту применяется логическая функция, т. е. предикат **IsEmpty (<выражение>)**, что буквально означает «есть пустой». Этот предикат обычно встроен в систему управления базами данных и может применяться к выражению абсолютно любого типа. Если такого предиката в системах управления базами данных нет, то можно написать логическую функцию самим и включить ее в список объектов проектируемой базы данных.

Рассмотрим еще один пример, когда не так просто определить, пустое ли мы имеем значение. Данные типа «дата». Какое значение в этом типе считать пустым значением, если дата может варьироваться в диапазоне от 01.01.0100. до 31.12.9999? Для этого в СУБД вводится специальное обозначение для **константы пустой даты {…}**, если значения этого типа записывается: {ДД. ММ. ГГ} или {ГГ. ММ. ДД}. С этим значением и происходит сравнение при проверке значения на пустоту. Оно считается вполне определенным, «полноправным» значением выражения этого типа, причем наименьшим из возможных.

При работе с базами данных пустые значения часто используются как значения по умолчанию или применяются, если значения выражений отсутствуют.

2. Неопределенные значения (Null-значения)

Слово **Null** используется для обозначения **неопределенных значений** в базах данных.

Чтобы лучше понять, какие значения понимаются под неопределенными, рассмотрим таблицу, являющуюся фрагментом базы данных:

№	Фамилия	Год рождения	№ паспорта
1	Хайретдинов	1980	Null
2	Карамазов	2000	Null
3	Коваленко	Null	Null

Итак, **неопределенное значение** или **Null-значение** – это:

1) неизвестное, но обычное, т. е. применимое значение. Например, у господина Хайретдинова, который является номером один в нашей базе данных, несомненно, имеются какие-то паспортные данные (как у человека 1980 г. рождения и гражданина страны), но они не известны, следовательно, не занесены в базу данных. Поэтому в соответствующую графу таблицы будет записано значение Null;

2) неприменимое значение. У господина Карамазова (№ 2 в нашей базе данных) просто не может быть никаких паспортных данных, потому что на момент создания этой базы данных или внесения в нее данных, он являлся ребенком;

3) значение любой ячейки таблицы, если мы не можем сказать применимое оно или нет. Например, у господина Коваленко, который занимает третью позицию в составленной нами базе данных, неизвестен год рождения, поэтому мы не можем с уверенностью говорить о наличие или отсутствии у него паспортных данных. А следовательно, значениями двух ячеек в строке, посвященной господину Коваленко будет Null-значение (первое – как неизвестное вообще, второе – как значение, природа которого неизвестна). Как и любые другие типы данных, Null-значения тоже имеют определенные **свойства**. Перечислим самые существенные из них:

1) с течением времени понимание Null-значения может меняться. Например, у господина Карамазова (№ 2 в нашей базе данных) в 2014 г., т. е. по достижении совершеннолетия, Null-значение изменится на какое-то конкретное вполне определенное значение;

2) Null-значение может быть присвоено переменной или константе любого типа (числового, строкового, логического, дате, времени и т. д.);

3) результатом любых операций над выражениями с Null-значе-ниями в качестве операндов является Null-значение;

4) исключением из предыдущего правила являются операции конъюнкции и дизъюнкции в условиях законов поглощения (подробнее о законах поглощения смотрите в п. 4 лекции № 2).

3. Null-значения и общее правило вычисления выражений

Поговорим подробнее о действиях над выражениями, содержащими Null-значения.

Общее правило работы с Null-значениями (то, что результат операций над Null-значениями есть Null-значение) применяется к следующим операциям:

1) к арифметическим;

2) к побитным операциям отрицания, конъюнкции и дизъюнкции (кроме законов поглощения);

3) к операциям со строками (например, конкотинации – сцепления строк);

4) к операциям сравнения (<, ≤, ≠, ≥, >).

Приведем примеры. В результате применений следующих операций будут получены Null-значения:

3 + Null, 1/ Null, (Иванов' + '' + Null) # Null

Здесь вместо обычного равенства использована **операция подстановки** «#» из-за особого характера работы с Null-значениями. Далее в подобных ситуациях также будет использоваться этот символ, который означает, что выражение справа от символа подстановки может заменить собой любое выражение из списка слева от символа подстановки.

Характер Null-значений приводит к тому, что часто в некоторых выражениях вместо ожидаемого нуля получается Null-значение, например:

(x – x), y * (x – x), x * 0 # Null при x = Null.

Все дело в том, что при подстановке, например, в выражение (x – x) значения x = Null, мы получаем выражение (Null – Null), и в силу вступает общее правило вычисления значения выражения, содержащего Null-значения, и информация о том, что здесь Null-значение соответствует одной и той же переменной теряется.

Можно сделать вывод, что при вычислении любых операций, кроме логических, Null-значения интерпретируются как **неприменимые**, и поэтому в результате получается тоже Null-значение.

К не менее неожиданным результатам приводит использование Null-значений в операциях сравнения. Например, в следующих выражениях также получаются Null-значения вместо ожидаемых логических значений True или False:

(Null < Null); (Null ≤ Null); (Null = Null); (Null ≠ Null);
(Null > Null); (Null ≥ Null) # Null;

Таким образом, делаем вывод, что нельзя говорить о том, что Null-значение равно или не равно самому себе. Каждое новое вхождение Null-значения рассматривается как независимое, и каждый раз Null-значения воспринимаются как различные неизвестные значения. Этим Null-значения кардинально отличаются от всех остальных типов данных, ведь мы знаем, что обо всех пройденных ранее величинах и их типах с уверенностью можно было говорить, что они равны или не равны друг другу.

Итак, мы видим, что Null-значения не являются значениями переменных в обычном смысле этого слова. Поэтому становится невозможным сравнивать значения переменных или выражения, содержащие Null-значения, поскольку в результате мы будем получать не логические значения True или False, а Null-значения, как в следующих примерах:

(x < Null); (x ≤ Null); (x = Null); (x ≠ Null); (x > Null);
(x ≥ Null) # Null;

Поэтому по аналогии с пустыми значениями для проверки выражения на Null-значения необходимо использовать специальный предикат:

IsNull (<выражение>), что буквально означает «есть Null».

Логическая функция возвращает значение True, если в выражении присутствует Null или оно равно Null, и False – в противном случае, но никогда не возвращает значение Null. Предикат IsNull может применяться к переменным и выражению любого типа. Если применять его к выражениям пустого типа, предикат всегда будет возвращать False.

Например:

IsNull(0)	False
IsNull(x + 'abc' + Null)	True
IsNull(2 * Null)	True
IsNull(Null)	True

Итак, действительно, видим, что в первом случае, когда предикат IsNull взяли от нуля, на выходе получилось значение False. Во всех случаях, в том числе во втором и третьем, когда аргументы логической функции оказались равными Null-значению, и в четвертом случае, когда сам аргумент и был изначально равен Null-значению, предикат выдал значение True.

4. Null-значения и логические операции

Обычно в системах управления базами данных непосредственно поддерживаются только три логические операции: отрицание ¬, конъюнкция & и дизъюнкция #. Операции следования # и равносильности # выражаются через них с помощью подстановок:

$(x \# y) \# (\neg x \# y)$;
$(x \# y) \# (x \# y) \ \& \ (y \# x)$;

Заметим, что эти подстановки полностью сохраняются и при использовании Null-значений.

Интересно, что при помощи операции отрицания «¬» любая из операций конъюнкция & или дизъюнкция # может быть выражена одна через другую следующим образом:

$(x \ \& \ y) \#\neg (\neg x \#\neg y)$;
$(x \# y) \# \neg (\neg x \ \& \ \neg y)$;

На эти подстановки, как и на предыдущие, Null-значения влияния не оказывают.

А теперь приведем таблицы истинности логических операций отрицания, конъюнкции и дизъюнкции, но кроме привычных значений True и False, используем также Null-значение в качестве операндов. Для удобства введем следующие обозначения: вместо True будем писать t, вместо False – f, а вместо Null – n.

1. **Отрицание** ¬x.

x	$\neg\ x$
f	t
n	n
t	f

Стоит отметить следующие интересные моменты касательно операции отрицания с использованием Null-значений:

1) ¬¬x # x – закон двойного отрицания;

2) ¬Null # Null – Null-значение является неподвижной точкой.

2. **Конъюнкция x & y**.

y / *x*	f	n	t
f	f	f	f
n	f	n	n
t	f	n	t

Эта операция также имеет свои свойства:
1) *x* & *y* # *y* & *x*– коммутативность;
2) x & x # x – идемпотентность;
3) False & *y* # False, здесь False – поглощающий элемент;
4) True & *y* # *y*, здесь True – нейтральный элемент.

3. **Дизъюнкция x # y.**

y / *x*	f	n	t
f	f	n	t
n	n	n	t
t	t	t	t

Свойства:
1) *x* # *y* # *y* # *x* – коммутативность;
2) *x* # *x* # *x* – идемпотентность;
3) False # *y* # *y*, здесь False – нейтральный элемент;
4) True # *y* # True, здесь True – поглощающий элемент.

Исключение из общего правила составляют правила вычисления логических операций конъюнкция & и дизъюнкция # в условиях действия **законов поглощения**:

(False & *y*) # (*x* & False) # False;
(True # *y*) # (*x* # True) # True;

Эти дополнительные правила формулируются для того, чтобы при замене Null-значения значениями False или True результат бы все равно не зависел бы от этого значения.

Как и ранее было показано для других типов операций, применение Null-значений в логических операциях могут также привести к неожиданным значениям. Например, логика на первый взгляд нарушена в **законе исключения третьего** (x # ¬x) и в **законе рефлексивности** (x = x), поскольку при x # Null имеем:

(x # ¬x), (x = x) # Null.

Законы не выполняются! Объясняется это так же, как и раньше: при подстановке Null-значения в выражение информация о том, что это значение сообщается одной и той же переменной теряется, а в силу вступает общее правило работы с Null-значениями.

Таким образом, делаем вывод: при выполнении логических операций с Null-значениями в качестве операнда эти значения определяются системами управления базами данных как **применимое, но неизвестное**.

5. Null-значения и проверка условий

Итак, из всего вышесказанного можно сделать вывод, что в логике систем управления базами данных имеются не два логических значения (True и False), а три, ведь Null-значение также рассматривается как одно из возможных логических значений. Именно поэтому на него часто ссылаются как на неизвестное значение, значение Unknown.

Однако, несмотря на это, в системах управления базами данных реализуется только двузначная логика. Поэтому условие с Null-значением (неопределенное условие) должно интерпретироваться машиной либо как True, либо как False.

В языке СУБД по умолчанию установлено опознавание условия с Null-значением как значения False. Проиллюстрируем это следующими примерами реализации в системах управления базами данных условных операторов If и While:

If P then A else B;

Эта запись означает: если P принимает значение True, то выполняется действие A, а если P принимает значение False или Null, то выполняется действие B.

Теперь применим к этому оператору операцию отрицания, получим:

If ¬P then B else A;

В свою очередь, этот оператор означает следующее: если ¬P принимает значение True, то выполняется действие B, а в том случае, если ¬P принимает значение False или Null, то будет выполняться действие A.

И снова, как мы видим, при появлении Null-значения мы сталкиваемся с неожиданными результатами. Дело в том, что два оператора If в этом примере не эквивалентны! Хотя один из них получен из другого отрицанием условия и перестановкой ветвей, т. е. стандартной операцией. Такие операторы в общем случае эквивалентны! Но в нашем примере мы видим, что Null-значению условия P в первом случае соответствует команда B, а во втором – A.

А теперь рассмотрим действие условного оператора While:

While P do A; B;

Как работает этот оператор? Пока переменная P имеет значение True, будет выполняться действие A, а как только P примет значение False или Null, выполнится действие B.

Но не всегда Null-значения интерпретируются как False. Например, в ограничениях целостности неопределенные условия опознаются как True (ограничения целостности – это условия, накладываемые на входные данные и обеспечивающие их корректность). Это происходит потому, что в таких ограничениях отвергнуть нужно только заведомо ложные данные.

И опять-таки в системах управления базами данных существует специальная **функция подмены IfNull (ограничения целостности, True)**, с помощью которой Null-значения и неопределенные условия можно представить в явном виде.

Перепишем условные операторы If и While с использованием этой функции:

1) If IfNull (P, False) then A else B;
2) While IfNull (P, False) do A; B;

Итак, функция подмены IfNull (выражение 1, выражение 2) возвращает значение первого выражения, если оно не содержит Null-значения, и значение второго выражения – в противном случае.

Надо заметить, что на тип возвращаемого функцией IfNull выражения никаких ограничений не накладывается. Поэтому с помощью этой функции можно явно переопределить любые правила работы с Null-значениями.

Лекция № 3. Реляционные объекты данных

1. Требования к табличной форме представления отношений

1. Самое первое требование, предъявляемое к табличной форме представления отношений, – это конечность. Работать с бесконечными таблицами, отношениями или любыми другими представлениями и организациями данных неудобно, редко оправдываются затраченные усилия, и, кроме того, подобное направление имеет малое практическое приложение.

Но помимо этого, вполне ожидаемого, существуют и другие требования.

2. Заголовок таблицы, представляющей отношение, должен обязательно состоять из одной строки – заголовка столбцов, причем с уникальными именами. Многоярусных заголовков не допускается. Например, таких:

A			B		C
1	2	3	1	2	
…	…	…	…	…	…

Все многоярусные заголовки заменяются одноярусными путем подбора подходящих заголовков. В нашем примере таблица после указанных преобразований будет выглядеть следующим образом:

A_1	A_2	A_3	B_1	B_2	C
…	…	…	…	…	…

Мы видим, что имя каждого столбца уникально, поэтому их можно как угодно менять местами, т. е. их порядок становится несущественным.

А это очень важно, поскольку является третьим свойством.

3. Порядок строк должен быть несущественным. Однако это требование также не является строго ограничительным, так как можно без труда привести любую таблицу к требуемому виду. Например, можно ввести дополнительный столбец, который будет определять порядок строк. В этом случае от перестановки строк тоже ничего не изменится. Вот пример такой таблицы:

...	...	**Порядок**
...	...	1
...	...	3
...	...	2

4. В таблице, представляющей отношение, не должно быть строк-дубликатов. Если же в таблице встречаются повторяющиеся строки, это можно легко исправить введением дополнительного столбца, отвечающего за количество дубликатов каждой строки, например:

...	...	**Число дубликатов**
...	...	0
...	...	3
...	...	1

Следующее свойство также является вполне ожидаемым, потому что лежит в основе всех принципов программирования и проектирования реляционных баз данных.

5. Данные во всех столбцах должны быть одного и того же типа. И кроме того они должны быть простого типа.

Поясним, что такое простой и сложный типы данных.

Простой тип данных – это такой тип, значения данных которого не являются составными, т. е. не содержат составных частей. Таким образом, в столбцах таблицы не должны присутствовать ни списки, ни массивы, ни деревья, ни подобные названным составные объекты.

Такие объекты – **составной тип данных** – в реляционных системах управления базами данных сами представляются в виде самостоятельных таблиц-отношений.

2. Домены и атрибуты

Домены и атрибуты – базовые понятия в теории создания баз данных и управления ими. Поясним, что же это такое.

Формально, **домен атрибута** (обозначается **dom(a)**), где a – некий атрибут, определяется как множество допустимых значений одного и того же типа соответствующего атрибута a. Этот тип должен быть простым, т. е:

$$dom(a) \# \{x \mid type(x) = type(a)\};$$

Атрибут (обозначается a), в свою очередь, определяется как упорядоченная пара, состоящая из имени атрибута name(a) и домена атрибута dom(a), т. е.:

$$a = (name(a): dom(a));$$

В этом определении вместо привычного знака «,» (как в стандартных определениях упорядоченных пар) используется «:». Это делается для того, чтобы подчеркнуть ассоциацию домена атрибута и типа данных атрибута.

Приведем несколько примеров различных атрибутов:

a_1 = (Курс: {1, 2, 3, 4, 5});
a_2 = (МассаКг: {x | type(x) = real, x 0});
a_3 = (ДлинаСм: {x | type(x) = real, x 0});

Заметим, что у атрибутов a_2 и a_3 домены формально совпадают. Но семантическое значение этих атрибутов различно, ведь сравнивать значения массы и длины бессмысленно. Поэтому домен атрибута ассоциируется не только с типом допустимых значений, но и семантическим значением.

В табличной форме представления отношений атрибут отображается как заголовок столбца таблицы, и при этом домен атрибута не указывается, но подразумевается. Это выглядит следующим образом:

a_1	a_2	a_3
...	...	...
...	...	...

Нетрудно заметить, что здесь каждый из заголовков a_1, a_2, a_3 столбцов таблицы, представляющей какое-то отношение, является отдельным атрибутом.

3. Схемы отношений. Именованные значения кортежей

В теории и практике СУБД понятия схемы отношения и именованного значения кортежа на атрибуте являются базовыми. Приведем их.

Схема отношения (обозначается **S**) определяется как конечное множество атрибутов с уникальными именами, т. е.:

$$S = \{a \mid a \# S\};$$

В каждой таблице, представляющей отношение, все заголовки столбцов (все атрибуты) объединяются в схему этого отношения.

Количество атрибутов в схеме отношений определяет **степень** этого **отношения** и обозначается как мощность множества: |**S**|.

Схема отношений может ассоциироваться с именем схемы отношений.

В табличной форме представления отношений, как нетрудно заметить, схема отношения – это не что иное, как строка заголовков столбцов.

a_1	a_2	a_3	a_4
...	...	...	...

$S = \{a_1, a_2, a_3, a_4\}$ – схема отношений этой таблицы.

Имя отношения изображается как схематический заголовок таблицы.

В текстовой же форме представления схема отношений может быть представлена как именованный список имен атрибутов, например:

Студенты (№ зачетной книжки, Фамилия, Имя, Отчество, Дата рождения).

Здесь, как и в табличной форме представления, домены атрибутов не указываются, но подразумеваются.

Из определения следует, что схема отношения может быть и пустой (S = #). Правда, возможно это только в теории, так как на практике система управления базами данных никогда не допустит создания пустой схемы отношения.

Именованное значение кортежа на атрибуте (обозначается **t(a)**)определяется по аналогии с атрибутом как упорядоченная пара, состоящая из имени атрибута и значения атрибута, т. е.:

$$t(a) = (name(a) : x), x \# dom(a);$$

Видим, что значение атрибута берется из домена атрибута.

В табличной форме представления отношения каждое именованное значение кортежа на атрибуте – это соответствующая ячейка таблицы:

...	...	...
$t(a_1)$	$t(a_2)$	$t(a_3)$
...	...	...

Здесь $t(a_1)$, $t(a_2)$, $t(a_3)$ – именованные значения кортежа t на атрибутах a_1, a_2, a_3.

Простейшие примеры именованных значений кортежей на атрибутах:

(Курс: 5), (Балл: 5);

Здесь соответственно Курс и Балл – имена двух атрибутов, а 5 – это одно из их значений, взятое из их доменов. Разумеется, хоть эти значения в обоих случаях равны друг другу, семантически они различны, так как множества этих значений в обоих случаях отличаются друг от друга.

4. Кортежи. Типы кортежей

Понятие кортежа в системах управления базами данных может быть интуитивно найдено уже из предыдущего пункта, когда мы говорили об именованном значении *кортежа* на различных атрибутах. Итак, **кортеж** (обозначается **t**, от англ. tuple – «кортеж») со схемой отношения S определяется как множество именованных значений этого кортежа на всех атрибутах, входящих в данную схему отношений S. Другими словами, атрибуты берутся из **области определения кортежа, def(t)**, т. е.:

$$t \equiv t(S) = \{t(a) \mid a \# def(t) \# S;.$$

Важно, что одному имени атрибута обязательно должно соответствовать не более одного значения атрибута.

В табличной форме записи отношения кортежем будет любая строка таблицы, т. е.:

...	...	...	...
$t(a_1)$	$t(a_2)$	$t(a_3)$	$t(a_4)$
$t(a_5)$	$t(a_6)$	$t(a_7)$	$t(a_8)$
...	...	...	...

Здесь $t_1(S) = \{t(a_1), t(a_2), t(a_3), t(a_4)\}$ и $t_2(S) = \{t(a_5), t(a_6), t(a_7), t(a_8)\}$ – кортежи.

Кортежи в СУБД различаются по **типам** в зависимости от своей области определения. Кортежи называются:

1) **частичными,** если их область определения включается или совпадает со схемой отношения, т. е. def(t) # S.

Это общий случай в практике баз данных;

2) **полными,** в том случае если их область определения полностью совпадает, равна схеме отношения, т. е. def(t) = S;

3) **неполными,** если область определения полностью включается в схему отношений, т. е. def(t) # S;

4) **нигде не определенными,** если их область определения равна пустому множеству, т. е. def(t) = #.

Поясним на примере. Пусть у нас имеется отношение, заданное следующей таблицей.

a	b	C
10	20	30
10	20	Null
Null	Null	Null

Пусть здесь t_1 = {10, 20, 30}, t_2 = {10, 20, Null}, t_3 = {Null, Null, Null}. Тогда легко заметить, что кортеж t_1 – полный, так как его область определения def(t_1) = { a, b, c} = S.

Кортеж t_2 – неполный, def(t_2) = { a, b} # S. И, наконец, кортеж t_3 – нигде не определенный, так как его def(t3) = #.

Надо заметить, что нигде не определенный кортеж – это пустое множество, тем не менее ассоциируемое со схемой отношений. Иногда нигде не определенный кортеж обозначается: #(S). Как мы уже видели в приведенном примере, такой кортеж представляет собой строку таблицы, состоящую только из Null-значений.

Интересно, что **сравнимыми**, т. е. возможно равными, являются только кортежи с одной и той же схемой отношений. Поэтому, например, два нигде не определенных кортежа с различными схемами отношений не будут равными, как могло ожидаться. Они будут различными так же, как их схемы отношений.

5. Отношения. Типы отношений

И наконец дадим определение отношению, как некой вершине пирамиды, состоящей из всех предыдущих понятий. Итак, **отношение** (обозначается **r**, от англ. relation – «отношение») со схемой отношений S определяется как обязательно конечное множество кортежей, имеющих ту же схему отношения S. Таким образом:

$$r \equiv r(S) = \{t(S) \mid t \# r\};$$

По аналогии со схемами отношений количество кортежей в отношении называют **мощностью отношений** и обозначают как мощность множества: |**r**|.

Отношения, как и кортежи, различаются по типам. Итак, отношения называются:

1) **частичными**, если для любого входящего в отношение кортежа выполняется следующее условие: [def(t) # S].

Это (как и с кортежами) общий случай;

2) **полными**, в том случае если #t # r(S) выполняется: [def(t) = S];

3) **неполными**, если #t # r(S) def(t) # S;

4) **нигде не определенными**, если #t # r(S) [def(t) = #].

Обратим отдельное внимание на нигде не определенные отношения. В отличие от кортежей работа с такими отношениями включает в себя небольшую тонкость. Дело в том, что нигде не определенные отношения могут быть двух видов: они могут быть либо пустыми, либо могут содержать единственный нигде не определенный кортеж (такие отношения обозначаются {#(S)}).

Сравнимыми (по аналогии с кортежами), т. е., возможно равными, являются лишь отношения с одной и той же схемой отношения. Поэтому отношения с различными схемами отношений являются различными.

В табличной форме представления, отношение – это тело таблицы, которому соответствует строка – заголовок столбцов, т. е. буквально – вся таблица, вместе с первой строкой, содержащей заголовки.

Лекция № 4. Реляционная алгебра. Унарные операции

Реляционная алгебра, как нетрудно догадаться, – это особая разновидность алгебры, в которой все операции производятся над реляционными моделями данных, т. е. над отношениями.

В табличных терминах отношение включает в себя строки, столбцы и строку – заголовок столбцов. Поэтому естественными унарными операциями являются операции выбора определенных строк или столбцов, а также смены заголовков столбцов – переименования атрибутов.

1. Унарная операция выборки

Первой унарной операцией, которую мы рассмотрим, является **операция выборки** – операция выбора строк из таблицы, представляющей отношение, по какому-либо принципу, т. е. выбор строк-кортежей, удовлетворяющих определенному условию или условиям.

Оператор выборки обозначается $\sigma<P>$, **условие выборки** – $P<S>$, т. е., оператор σ берется всегда с определенным условием на кортежи P, а само условие P записывается зависящим от схемы отношения S. С учетом всего этого сама **операция выборки** над схемой отношения S применительно к отношению r будет выглядеть следующим образом:

$$\sigma<P>r(S) \equiv \sigma<P>r = \{t(S) \mid t \# r \ \&\ P<S>t\} = \{t(S) \mid t \# r \ \&\ IfNull(P<S>t, False\};$$

Результатом этой операции будет новое отношение с той же схемой отношения S, состоящее из тех кортежей $t(S)$ исходного отношения-операнда, которые удовлетворяют условию выборки $P<S>t$. Понятно, что для того, чтобы применить какое-то условие к кортежу, необходимо подставить значения атрибутов кортежа вместо имен атрибутов.

Чтобы лучше понять принцип работы этой операции, приведем пример. Пусть дана следующая схема отношения:

S: Сессия (№ зачетной книжки, Фамилия, Предмет, Оценка).

Условие выборки возьмем такое:

$P<S>$ = (Предмет = ‘Информатика’ and Оценка > 3).

Нам необходимо из исходного отношения-операнда выделить те кортежи, в которых содержится информация о студентах, сдавших предмет «Информатика» не ниже, чем на три балла.

Пусть также дан следующий кортеж из этого отношения:

$t_0(S) \# r(S)$: {(№ зачетной книжки: 100), (Фамилия: ‘Иванов’), (Предмет: ‘Базы данных’), (Оценка: 5)};

Применяем наше условие выборки к кортежу t_0, получаем:

$P<S>t_0$ = (‘Базы данных’ = ‘Информатика’ and 5 > 3);

На данном конкретном кортеже условие выборки не выполняется.

А вообще результатом этой конкретной выборки

σ<Предмет = 'Информатика' and Оценка > 3 > Сессия

будет таблица «Сессия», в которой оставлены строки, удовлетворяющие условию выборки.

2. Унарная операция проекции

Еще одна стандартная унарная операция, которую мы изучим, – это операция проекции. **Операция проекции** – это операция выбора столбцов из таблицы, представляющей отношение, по какому-либо признаку. А именно машина выбирает те атрибуты (т. е. буквально те столбцы) исходного отношения-операнда, которые были указаны в проекции.

Оператор проекции обозначается $[S']$ или $\pi<S'>$. Здесь S' – подсхема исходной схемы отношения S, т. е. ее некоторые столбцы. Что это означает? Это означает, что у S' атрибутов меньше, чем у S, потому что в S' остались только те из них, для которых выполнилось условие проекции. А в таблице, представляющей отношение $r(S')$, строк столько же, сколько их у таблицы $r(S)$, а столбцов – меньше, так как остались только соответствующие оставшимся атрибутам. Таким образом, оператор проекции $\pi< S'>$ применительно к отношению $r(S)$ дает в результате новое отношение с другой схемой отношения $r(S')$, состоящее из проекций $t(S)$ $[S']$ кортежей исходного отношения. Как определяются эти проекции кортежей? **Проекция** любого кортежа $t(S)$ исходного отношения $r(S)$ на подсхему S' определяется следующей формулой:

$$t(S)\ [S'] = \{t(a) | a \# def(t) \cap S'\},\ S' \# S.$$

Важно заметить, что дубликаты кортежей из результата исключаются, т. е. в таблице, представляющей новое, результирующее отношение повторяющихся строк не будет.

С учетом всего вышесказанного, операция проекции в терминах систем управления базами данных будет выглядеть следующим образом:

$$\pi<S'>r(S) \equiv \pi<S'>r \equiv r(S)\ [S'] \equiv r\ [S'] = \{t(S)\ [S'] \mid t \# r\};$$

Рассмотрим пример, иллюстрирующий принцип работы операции выборки.

Пусть дано отношение «Сессия» и схема этого отношения:

S: Сессия (№ зачетной книжки, Фамилия, Предмет, Оценка);

Нас будут интересовать только два атрибута из этой схемы, а именно «№ зачетной книжки» и «Фамилия» студента, поэтому подсхема S' будет выглядеть следующим образом:

S' : (№ зачетной книжки, Фамилия).

Нужно исходное отношение $r(S)$ спроецировать на подсхему S'.

Далее, пусть нам дан кортеж $t_0(S)$ из исходного отношения:

$t_0(S) \# r(S)$: {(№ зачетной книжки: 100), (Фамилия: ‘Иванов’), (Предмет: ‘Базы данных’), (Оценка: 5)};

Значит, проекция этого кортежа на данную подсхему S' будет выглядеть следующим образом:

$t_0(S)\ S'$: {(№ зачетной книжки: 100), (Фамилия: ‘Иванов’)};

Если говорить об операции проекции в терминах таблиц, то проекция Сессия [№ зачетной книжки, Фамилия] исходного отношения – это таблица Сессия, из которой вычеркнуты все столбцы, кроме двух: № зачетной книжки и Фамилия. Кроме того, все дублирующиеся строки также удалены.

3. Унарная операция переименования

И последняя унарная операция, которую мы рассмотрим, – это **операция переименования атрибутов**. Если говорить об отношении как о таблице, то операция переименования нужна для того, чтобы поменять названия всех или некоторых столбцов.

Оператор переименования выглядит следующим образом: $\rho<\varphi>$, здесь φ — **функция переименования**.

Эта функция устанавливает взаимно-однозначное соответствие между именами атрибутов схем S и $\hat{S}$, где соответственно S — схема исходного отношения, а $\hat{S}$ — схема отношения с переименованными атрибутами. Таким образом, оператор $\rho<\varphi>$ в применении к отношению $r(S)$ дает новое отношение со схемой $\hat{S}$, состоящее из кортежей исходного отношения только с переименованными атрибутами.

Запишем операцию переименования атрибутов в терминах систем управления базами данных:

$$\rho<\varphi>\ r(S) \equiv \rho<\varphi> r = \{\rho<\varphi>\ t(S) |\ t \# r\};$$

Приведем пример использования этой операции:

Рассмотрим уже знакомое нам отношение Сессия, со схемой:

S: Сессия (№ зачетной книжки, Фамилия, Предмет, Оценка);

Введем новую схему отношения $\hat{S}$, с другими именами атрибутов, которые мы бы хотели видеть вместо имеющихся:

$\hat{S}$ *:* (№ ЗК, Фамилия, Предмет, Балл);

Например, заказчик базы данных захотел в вашем готовом отношении видеть другие названия. Чтобы воплотить в жизнь этот заказ, необходимо спроектировать следующую функцию переименования:

φ : (№ зачетной книжки, Фамилия, Предмет, Оценка) → (№ ЗК, Фамилия, Предмет, Балл);

Фактически, требуется поменять имя только у двух атрибутов, поэтому законно будет записать следующую функцию переименования вместо имеющейся:

φ : (№ зачетной книжки, Оценка) → (№ ЗК, Балл);

Далее, пусть дан также уже знакомый нам кортеж принадлежащий отношению Сессия:

$t_0(S)$ # $r(S)$*:* {(№ зачетной книжки: 100), (Фамилия: ‘Иванов’), (Предмет: ‘Базы данных’), (Оценка: 5)};

Применим оператор переименования к этому кортежу:

$\rho<\varphi>$ $t_0(S)$*:* {(№ ЗК: 100), (Фамилия: ‘Иванов’), (Предмет: ‘Базы данных’), (Балл: 5)};

Итак, это один из кортежей нашего отношения, у которого переименовали атрибуты.

В табличных терминах отношение

ρ < № зачетной книжки, Оценка → «№ ЗК, Балл > Сессия —

это новая таблица, полученная из таблицы отношения «Сессия», переименованием указанных атрибутов.

4. Свойства унарных операций

У унарных операций, как и у любых других, есть определенные свойства. Рассмотрим наиболее важные из них.

Первым свойством унарных операций выборки, проекции и переименования является свойство, характеризующее соотношение мощностей отношений. (Напомним, что мощность – это количество кортежей в том или ином отношении.) Понятно, что здесь рассматривается соответственно отношение исходное и отношение, полученное в результате применения той или иной операции.

Заметим, что все свойства унарных операций следуют непосредственно из их определений, поэтому их можно легко объяснить и даже при желании вывести самостоятельно.

Итак:

1) **соотношение мощностей:**

а) для операции выборки: $| \sigma<P>r | \leq |r|$;

б) для операции проекции: $| r[S'] | \leq |r|$;

в) для операции переименования: $| \rho<\varphi>r | = |r|$;

Итого, мы видим, что для двух операторов, а именно для оператора выборки и оператора проекции, мощность исходных отношений – операндов больше, чем мощность отношений, получаемых из исходных применением соответствующих операций. Это происходит потому, что при выборе, сопутствующему действию этих двух операций выборки и проекции, происходит исключение некоторых строк или столбцов, не удовлетворивших условиям выбора. В том случае, когда условиям удовлетворяют все строки или столбцы, уменьшения мощности (т. е. количества кортежей) не происходит, поэтому в формулах неравенство нестрогое.

В случае же операции переименования, мощность отношения не изменяется, за счет того, что при смене имен никакие кортежи из отношения не исключаются;

2) **свойство идемпотентности:**

а) для операции выборки: $\sigma<P>\ \sigma<P>r = \sigma<P>$;

б) для операции проекции: $r\ [S']\ [S'] = r\ [S']$;

в) для операции переименования в общем случае свойство идемпотентности неприменимо.

Это свойство означает, что двойное последовательное применение одного и того же оператора к какому-либо отношению равносильно его однократному применению.

Для операции переименования атрибутов отношения, вообще говоря, это свойство может быть применено, но обязательно со специальными оговорками и условиями.

Свойство идемпотентности очень часто используется для упрощения вида выражения и приведения его к более экономичному, актуальному виду.

И последнее свойство, которое мы рассмотрим, – это свойство монотонности. Интересно заметить, что при любых условиях все три оператора монотонны;

3) **свойство монотонности:**

а) для операции выборки: $r_1 \# r_2 \# \sigma<P>\ r_1 \# \sigma <P>r_2$;

б) для операции проекции: $r_1 \# r_2 \# r_1[S'] \# r_2\ [S']$;

в) для операции переименования: $r_1 \# r_2 \# \rho<\varphi>r_1 \# \rho <\varphi>r_2$;

Понятие монотонности в реляционной алгебре аналогично этому же понятию из алгебры обычной, общей. Поясним: если изначально отношения r_1 и r_2 были связаны между собой таким образом, что $r \# r_2$, то и после применения любого их трех операторов выборки, проекции или переименования это соотношение сохранится.

Лекция № 5. Реляционная алгебра. Бинарные операции

1. Операции объединения, пересечения, разности

У любых операций есть свои правила применимости, которые необходимо соблюдать, чтобы выражения и действия не теряли смысла. Бинарные теоретико-множественные операции объединения, пересечений и разности могут быть применены только к двум отношениям обязательно с одной и той же схемой отношения. Результатом таких бинарных операций будут являться отношения, состоящие из кортежей, удовлетворяющих условиям операций, но с такой же схемой отношения, как и у операндов.

1. Результатом **операции объединения** двух отношений $r_1(S)$ и $r_2(S)$ будет новое отношение $r_3(S)$, состоящее из тех кортежей отношений $r_1(S)$ и $r_2(S)$, которые принадлежат хотя бы одному из исходных отношений и с такой же схемой отношения.

Таким образом, пересечение двух отношений – это:

$$r_3(S) = r_1(S) \# r_2(S) = \{t(S) \mid t \# r_1 \# t \# r_2\};$$

Для наглядности, приведем пример в терминах таблиц:

Пусть даны два отношения:

$r_1(S)$:

S	
a	1
b	2

$r_2(S)$:

S	
b	2
c	3
d	4

Мы видим, что схемы первого и второго отношений одинаковы, только имеют различной количество кортежей. Объединением этих двух отношений будет отношение $r_3(S)$, которому будет соответствовать следующая таблица:

$r3(S) = r_1(S) \# r_2(S)$:

S	
a	1
b	2
c	3
d	4

Итак, схема отношения S не изменилась, только выросло количество кортежей.

2. Перейдем к рассмотрению следующей бинарной операции – **операции пересечения** двух отношений. Как мы знаем еще из школьной геометрии, в результирующее отношение войдут только те кортежи исходных отношений, которые присутствуют одновременно в обоих отношениях $r_1(S)$ и $r_2(S)$ (снова обращаем внимание на одинаковую схему отношения).

Операция пересечения двух отношений будет выглядеть следующим образом:

$$r_4(S) = r_1(S) \cap r_2(S) = \{t(S) \mid t \# r_1 \ \& \ t \# r_2\};$$

И снова рассмотрим действие этой операции над отношениями, представленными в виде таблиц:

$r_1(S)$:

S	
a	1
b	2

$r_2(S)$:

S	
b	2
c	3
d	4

Согласно определению операции пересечением отношений $r_1(S)$ и $r_2(S)$ будет новое отношение $r_4(S)$, табличное представление которого будет выглядеть следующим образом:

$r_4(S) = r_1(S) \cap r_2(S)$:

S	
b	2

Действительно, если посмотреть на кортежи первого и второго исходного отношений, общий среди них только один: {b, 2}. Он и стал единственным кортежем нового отношения $r_4(S)$.

3. **Операция разности** двух отношений определяется аналогичным с предыдущими операциями образом. Отношения-операнды, так же, как и в предыдущих операциях, должны иметь одинаковые схемы отношения, тогда в результирующее отношение войдут все те кортежи первого отношения, которых нет во втором, т. е.:

$$r_5(S) = r_1(S) \setminus r_2(S) = \{t(S) \mid t \# r_1 \ \& \ t \# r_2\};$$

Уже хорошо знакомые нам отношения $r_1(S)$ и $r_2(S)$, в табличном представлении выглядящие следующим образом:

$r_1(S)$:

S	
a	1
b	2

$r_2(S)$:

S	
b	2
c	3
d	4

Мы рассмотрим как операнды в операции пересечения двух отношений. Тогда, следуя данному определению, результирующее отношение $r5(S)$ будет выглядеть следующим образом:

$r_5(S) = r_1(S) \setminus r_2(S)$:

S	
a	1

Рассмотренные бинарные операции являются базовыми, на них основываются другие операции, более сложные.

2. Операции декартового произведения и естественного соединения

Операция декартового произведения и операция естественного соединения являются бинарными операциями типа произведения и основываются на операции объединения двух отношений, которую мы рассматривали ранее.

Хотя действие операции декартова произведения многим может показаться знакомым, начнем мы все-таки с операции естественного произведения, так как она является более общим случаем, нежели первая операция.

Итак, рассмотрим операцию естественного соединения. Следует сразу заметить, что операндами этого действия могут являться отношения с разными схемами в отличие от трех бинарных операций объединения, пересечения и переименования.

Если рассмотреть два отношения с различными схемами отношений $r_1(S_1)$ и $r_2(S_2)$, то их **естественным соединением** будет новое отношение $r_3(S_3)$, которое будет состоять только из тех кортежей операндов, которые совпадают на пересечении схем отношений. Соответственно, схема нового отношения будет больше любой из схем отношений исходных, так как является их соединением, «склеиванием». Кстати, кортежи, одинаковые в двух отношениях-операндах, по которым и происходит это «склеивание», называются **соединимыми**.

Запишем определение операции естественного соединения на языке формул систем управления базами данных:

$$r_3(S_3) = r_1(S_1) \times r_2(S_2) = \{t(S_1 \# S_2) \mid t[S_1] \# r_1 \ \& \ t(S_2) \# r_2\};$$

Рассмотрим пример, хорошо иллюстрирующий работу естественного соединения, его «склеивание». Пусть дано два отношения $r_1(S_1)$ и $r_2(S_2)$, в табличной форме представления соответственно равные:

$r_1(S_1)$:

a	1
b	1
c	3

$r_2(S_2)$:

1	x
2	y
3	z

Мы видим, что у этих отношений присутствуют кортежи, совпадающие при пересечении схем S_1 и S_2 отношений. Перечислим их:

1) кортеж {a, 1} отношения $r_1(S_1)$ совпадает с кортежем {1, x} отношения $r_2(S_2)$;

2) кортеж {b, 1} из $r_1(S_1)$ также совпадает с кортежем {1, x} из $r_2(S_2)$;

3) кортеж {c, 3} совпадает с кортежем {3, z}.

Значит, при естественном соединении новое отношение $r_3(S_3)$ получается «склеиванием» именно на этих кортежах. Таким образом, $r_3(S_3)$ в табличном представлении будет выглядеть следующим образом:

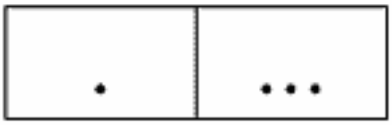
$r_3(S_3) = r_1(S_1) \times r_2(S_2)$:

a	1	x
b	2	y
c	3	z

Получается по определению: схема S_3 не совпадает ни со схемой S_1, ни со схемой S_2, мы «склеили» две исходные схемы по пересекающимся кортежам, чтобы получить их естественное соединение.

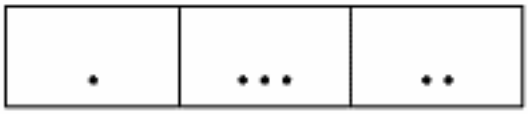
Покажем схематично, как происходит соединение кортежей при применении операции естественного соединения.

Пусть отношение r_1 имеет условный вид:



А отношение r_2 – вид:

Тогда их естественное соединение будет выглядеть следующим образом:



Видим, что «склеивание» отношений-операндов происходит по той самой схеме, что мы приводили ранее, рассматривая пример.

Операция **декартового соединения** является частным случаем операции естественного соединения. Если конкретнее, то, рассматривая действие операции декартового произ-

ведения на отношения, мы заведомо оговариваем, что в этом случае может идти речь только о *непересекающихся* схемах отношений. В результате применения обеих операций получаются отношения со схемами, равными объединению схем отношений-операндов, только в декартово произведение двух отношений попадают всевозможные пары их кортежей, так как схемы операндов ни в коем случае не должны пересекаться.

Таким образом, исходя из всего вышесказанного запишем математическую формулу для операции декартового произведения:

$$r_4(S_4) = r_1(S1) \times r_2(S_2) = \{t(S_1 \# S_2) \mid t\,[S_1] \# r_1 \;\&\; t(S_2) \# r_2\},\ S_1 \cap S_2 = \#;$$

Теперь рассмотрим пример, чтобы показать, какой вид будет иметь результирующая схема отношения, при применении операции декартового произведения.

Пусть даны два отношения $r_1(S1)$ и $r_2(S_2)$, которые в табличном виде представляются следующим образом:

$r_1(S_1)$:

a
b
c

$r_2(S_2)$:

x
y

Итак, мы видим, что ни один из кортежей отношений $r_1(S_1)$ и $r_2(S_2)$, действительно, не совпадает в их пересечении. Поэтому в результирующее отношение $r_4(S_4)$ попадут всевозможные пары кортежей первого и второго отношений-операндов. Получится:

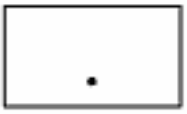
$r_4(S_4) = r_1(S1) \times r_2(S_2)$:

a	x
a	y
b	x
b	y
c	x
c	y

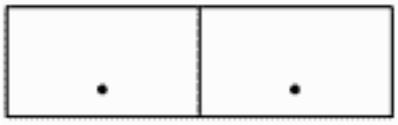
Получилась новая схема отношения $r_4(S_4)$ не «склеиванием» кортежей как в предыдущем случае, а перебором всех возможных различных пар несовпадающих в пересечении исходных схем кортежей.

Снова, как и в случае естественного соединения, приведем схематичный пример работы операции декартового произведения.

Пусть r_1 задано следующим условным образом:



А отношение r_2 задано:



Тогда их декартовое произведение схематично можно изобразить следующим образом:

Именно таким образом и получается результирующее отношение при применении операции декартового произведения.

3. Свойства бинарных операций

Из приведенных выше определений бинарных операций объединения, пересечения, разности, декартового произведения и естественного соединения следуют свойства.

1. Первое свойство, как и в случае унарных операций, иллюстрирует **соотношение мощностей** отношений:

1) для операции объединения:

$$|r_1 \# r_2| \le |r_1| + |r_2|;$$

2) для операции пересечения:

$$|r_1 \cap r_2| \le min(|r_1|, |r_2|);$$

3) для операции разности:

$$|r_1 \setminus r_2| \le |r_1|;$$

4) для операции декартового произведения:

$$|r_1 \times r_2| = |r_1| \cdot |r_2|;$$

5) для операции естественного соединения:

$$|r_1 \times r_2| \le |r_1| \cdot |r_2|.$$

Соотношение мощностей, как мы помним, характеризует, как меняется количество кортежей в отношениях после применения той или иной операции. Итак, что мы видим? Мощность **объединения** двух отношений r_1 и r_2 меньше суммы мощностей исходных отношений-операндов. Почему это происходит? Все дело в том, что при объединении совпадающие кортежи исчезают, накладываясь друг на друга. Так, обратившись к примеру, который мы рассматривали по прохождении этой операции, можно заметить, что в первом отношении было два кортежа, во втором – три, а в результирующем – четыре, т. е. меньше, чем пять (сумма мощностей отношений-операндов). По совпадающему кортежу {b, 2} эти отношения «склеились».

Мощность результата **пересечения** двух отношений меньше или равна минимальной мощности исходных отношений-операндов. Обратимся к определению этой операции: в результирующее отношение попадают только те кортежи, которые присутствуют в обоих отношениях исходных. А значит, мощность нового отношения никак не может превышать мощности того отношения-операнда, число кортежей которого наименьшее из двух. А равной этой минимальной мощности мощность результата быть может, так как всегда допускается случай, когда все кортежи отношения с меньшей мощностью совпадают с какими-то кортежами второго отношения-операнда.

В случае операции **разности** все достаточно тривиально. Действительно, если из первого отношения-операнда «вычесть» все кортежи, присутствующие также во втором отношении, то их количество (а следовательно, мощность) уменьшится. В том случае, если ни один кортеж первого отношения не совпадет ни с одним кортежем отношения второго, т. е. «вычитать» будет нечего, мощность его не уменьшится.

Интересно, что в случае применения операции **декартового произведения** мощность результирующего отношения в точности равна произведению мощностей двух отношений-операндов. Понятно, что это происходит потому, что в результат записываются все возможные пары кортежей исходных отношений, а ничего не исключается.

И, наконец, операцией **естественного соединения** получается отношение, мощность которого больше или равна произведения мощностей двух исходных отношений. Опять-таки

это происходит потому, что отношения-операнды «склеиваются» по совпадающим кортежам, а несовпадающие – из результата исключаются вовсе.

2. **Свойство идемпотентности:**

1) для операции объединения: $r \# r = r$;

2) для операции пересечения: $r \cap r = r$;

3) для операции разности: $r \setminus r \neq r$;

4) для операции декартового произведения (в общем случае, свойство не применимо);

5) для операции естественного соединения: $r \times r = r$.

Интересно, что свойство идемпотентности верно не для всех операций из приведенных, а для операции декартового произведения оно и вовсе не применимо. Действительно, если объединить, пересечь или естественно соединить какое-либо отношение само с собой, оно не изменится. А вот если отнять от отношения точно равное ему отношение, в результате получится пустое отношение.

3. **Свойство коммутативности:**

1) для операции объединения:

$$r_1 \# r_2 = r_2 \# r_1;$$

2) для операции пересечения:

$$r \cap r = r \cap r;$$

3) для операции разности:

$$r_1 \setminus r_2 \neq r_2 \setminus r_1;$$

4) для операции декартового произведения:

$$r_1 \times r_2 = r_2 \times r_1;$$

5) для операции естественного соединения:

$$r_1 \times r_2 = r_2 \times r_1.$$

Свойство коммутативности выполняется для всех операций, кроме операции разности. Это легко понять, ведь от перестановки отношений местами их состав (кортежи) не меняется. А при применении операции разности важно, какое из отношений-операндов стоит на первом месте, потому что от этого зависит, кортежи какого отношения примутся за эталонные, т. е. с какими кортежами будут сравниваться другие кортежи на предмет исключения.

4. **Свойство ассоциативности:**

1) для операции объединения:

$$(r_1 \# r_2) \# r_3 = r_1 \#(r_2 \# r_3);$$

2) для операции пересечения:

$$(r_1 \cap r_2) \cap r_3 = r_1 \cap (r_2 \cap r_3);$$

3) для операции разности:

$$(r_1 \setminus r_2) \setminus r_3 \neq r_1 \setminus (r_2 \setminus r_3);$$

4) для операции декартового произведения:

$$(r_1 \times r_2) \times r_3 = r_1 \times (r_2 \times r_3);$$

5) для операции естественного соединения:

$$(r_1 \times r_2) \times r_3 = r_1 \times (r_2 \times r_3).$$

И снова мы видим, что свойство выполняется для всех операций, кроме операции разности. Объясняется это таким же образом, как и в случае применения свойства коммута-

тивности. По большому счету, операциям объединения, пересечения, разности и естественного соединения все равно в каком порядке стоят отношения-операнды. Но при «отнимании» отношений друг от друга порядок играет главенствующую роль.

На основании вышеприведенных свойств и рассуждений можно сделать следующий вывод: три последних свойства, а именно свойство идемпотентности, коммутативности и ассоциативности, верны для всех рассмотренных нами операций, кроме операции разности двух отношений, для которой не выполнилось вообще ни одно из трех означенных свойств, и только в одном случае свойство оказалось неприменимым.

4. Варианты операций соединения

Используя как основу рассмотренные ранее унарные операции выборки, проекции, переименования и бинарные операции объединения, пересечения, разности, декартова произведения и естественного соединения (все они в общем случае называются **операциями соединения**), мы можем ввести новые операции, выведенные с помощью перечисленных понятий и определений. Подобная деятельность называется составлением **вариантов операций соединения**.

Первым таким вариантом операций соединения является операция **внутреннего соединения** по заданному условию соединения.

Операция внутреннего соединения по какому-то определенному условию определяется как производная операция от операций декартового произведения и выборки.

Запишем формульное определение этой операции:

$$r_1(S_1) \times_P r_2(S_2) = \sigma_{<P>}(r_1 \times r_2),\ S_1 \cap S_2 = \#;$$

Здесь $P = P <S_1 \# S_2>$ – условие, накладываемое на объединение двух схем исходных отношений-операндов. Именно по этому условию и происходит отбор кортежей из отношений r_1 и r_2 в результирующее отношение.

Следует отметить, что операция внутреннего соединения может применяться к отношениям с разными схемами отношений. Эти схемы могут быть любыми, но они ни в коем случае не должны пересекаться.

Кортежи исходных отношений-операндов, попавшие в результат операции внутреннего соединения, называются **соединимыми кортежами**.

Для наглядного иллюстрирования работы операции внутреннего соединения, приведем следующий пример.

Пусть нам даны два отношения $r_1(S_1)$ и $r_2(S_2)$ с различными схемами отношения:

$r_1(S_1)$:

a1	b1
a	1
b	1
c	3
d	4

$r_2(S_2)$:

b2	c2
1	x
2	y
3	z

Следующая таблица даст результат применения операции внутреннего соединения по условию P = (b1 = b2).

$r_1(S_1) \times_P r_2(S_2)$:

a1	b1	b2	c2
a	1	1	x
b	1	1	x
c	3	3	z

Итак, мы видим, что действительно «слипание» двух таблиц, представляющих отношения, произошло именно по тем кортежам, в которых выполняется условие операции внутреннего соединения P = (b1 = b2).

Теперь на основании уже введенной операции внутреннего соединения мы можем ввести операцию **левого внешнего соединения** и **правого внешнего соединения**. Поясним.

Результатом операции левое внешнее соединение является результат внутреннего соединения, пополненный несоединимыми кортежами левого исходного отношения-операнда. Аналогично результат операции правого внешнего соединения определяется как результат операции внутреннего соединения, пополненный несоединимыми кортежами стоящего справа исходного отношения-операнда.

Вопрос, чем же пополняются результирующие отношения операций левого и правого внешнего соединения, вполне ожидаем. Кортежи одного отношения-операнда дополняются на схеме другого отношения-операнда **Null-значениями**.

Стоит заметить, что введенные таким образом операции левого и правого внешнего соединения являются производными операциями от операции внутреннего соединения.

Чтобы записать общие формулы для операций левого и правого внешнего соединений, проведем некоторые дополнительные построения.

Пусть нам даны два отношения $r_1(S_1)$ и $r_2(S_2)$ с различными схемами отношений S_1 и S_2, не пересекающимися друг с другом.

Так как мы уже оговаривали, что операции левого и правого внутреннего соединения являются производными, то мы можем получить следующие вспомогательные формулы для определения операции левого внешнего соединения:

1) $r_3\ (S_2\ \#\ S_1)\ \#\ r_1(S_1) \times {}_P r_2(S_2)$;

$r_3\ (S_2\ \#\ S_1)$ — это просто результат внутреннего соединения отношений $r_1(S_1)$ и $r_2(S_2)$. Левое внешнее соединение является производной операцией именно от операции внутреннего соединения, поэтому мы и начинаем наши построения с нее;

2) $r_4(S_1)\ \#\ r_3(S_2\ \#S_1)\ [S_1]$;

Таким образом, с помощью унарной операции проекции, мы выделили все соединимые кортежи левого исходного отношения-операнда $r_1(S_1)$. Результат обозначили $r_4(S_1)$ для удобства применения;

3) $r_5\ (S_1)\ \#\ r_1(S_1) \setminus r_4(S_1)$;

Здесь $r_1(S_1)$ — все кортежи левого исходного отношения-операнда, а $r_4(S_1)$ – его же кортежи, только соединимые. Таким образом, при помощи бинарной операции разности, в отношении $r_5(S_1)$ у нас получились все несоединимые кортежи левого отношения-операнда;

4) $r_6(S_2)\#\ \{\#(S_2)\}$;

$\{\#(S_2)\}$ — это новое отношение со схемой (S_2), содержащее всего один кортеж, причем составленный из Null-значений. Для удобства мы обозначили это отношение $r_6(S_2)$;

5) $r_7\ (S_2\ \#\ S_1)\ \#\ r_5(S_1) \times r_6(S_2)$;

Здесь мы взяли полученные в пункте три, несоединимые кортежи левого отношения-операнда ($r_5(S_1)$) и дополнили их на схеме второго отношения-операнда S_2 Null-значениями, т. е. декартово умножили отношение, состоящее из этих самых несоединимых кортежей на отношение $r_6(S_2)$, определенное в пункте четыре;

6) $r_1(S1) \rightarrow\times {}_P\ r_2(S_2)\ \#\ (r_1 \times {}_P\ r_2)\ \#\ r_7\ (S_2\ \#\ S_1)$;

Это и есть **левое внешнее соединение**, полученное, как можно видеть, объединением декартового произведения исходных отношений-операндов r_1 и r_2 и отношения $r_7\ (S_2\ \#\ S_1)$, определенного в пункте пятом.

Теперь у нас имеются все необходимые выкладки для определения не только операции левого внешнего соединения, но по аналогии и для определения операции правого внешнего соединения. Итак:

1) операция **левого внешнего соединения** в строгом формулярном виде выглядит следующим образом:

$$r_1(S_1) \rightarrow\times {}_P\ r_2(S_2)\ \#\ (r_1 \times {}_P\ r_2)\ \#\ [(r_1 \setminus (r_1 \times {}_P\ r_2)\ [S_1]) \times \{\#(S_2)\}];$$

2) операция **правого внешнего соединения** определяется подобным образом операции левого внешнего соединения и имеет следующий вид:

$$r_1(S_1) \rightarrow\times {}_P\ r_2(S_2)\ \#\ (r_1 \times {}_P\ r_2)\ \#\ [(r_2 \setminus (r_1 \times {}_P\ r_2)\ [S_2]) \times \{\#(S_1)\}];$$

Эти две производные операции имеют всего два свойства, достойные упоминания.

1. Свойство коммутативности:

1) для операции левого внешнего соединения:

$$r_1(S_1) \rightarrow\times_P r_2(S_2) \neq r_2(S_2) \rightarrow\times_P r_1(S_1);$$

2) для операции правого внешнего соединения:

$$r_1(S_1) \leftarrow\times_P r_2(S_2) \neq r_2(S_2) \leftarrow\times_P r_1(S_1)$$

Итак, мы видим, что свойство коммутативности не выполняется для этих операций в общем виде, но при этом операции левого и правого внешнего соединения взаимно обратны друг другу, т. е. выполняется:

1) для операции левого внешнего соединения:

$$r_1(S_1) \rightarrow\times_P r_2(S_2) = r_2(S_2) \rightarrow\times_P r_1(S_1);$$

2) для операции правого внешнего соединения:

$$r_1(S_1) \leftarrow\times_P r_2(S_2) = r_2(S_2) \leftarrow\times_P r_1(S_1).$$

2. Основным свойством операций левого и правого внешнего соединения является то, что они позволяют **восстановить** исходное отношение-операнд по конечному результату той или иной операции соединения, т. е. выполняются:

1) для операции левого внешнего соединения:

$$r_1(S1) = (r_1 \rightarrow\times_P r_2)\ [S_1];$$

2) для операции правого внешнего соединения:

$$r_2(S_2) = (r_1 \leftarrow\times_P r_2)\ [S_2].$$

Таким образом, мы видим, что первое исходное отношение-операнд можно восстановить из результата операции левого правого соединения, а если конкретнее, то применением к результату этого соединения ($r_1 \times r_2$) унарной операции проекции на схему S_1, $[S_1]$.

И аналогично второе исходное отношение-операнд можно восстановить применением к результату операции правого внешнего соединения ($r_1 \times r_2$) унарной операции проекции на схему отношения S_2.

Приведем пример для более подробного рассмотрения работы операций левого и правого внешних соединений. Введем уже знакомые нам отношения $r_1(S_1)$ и $r_2(S_2)$ с различными схемами отношения:

$r_1(S_1)$:

a1	b1
a	1
b	1
c	3
d	4

$r_2(S_2)$:

b2	c2
1	x
2	y
3	z

Несоединимый кортеж левого отношения-операнда $r_2(S_2)$ – это кортеж {d, 4}. Следуя определению, именно им следует дополнить результат внутреннего соединения двух исходных отношений-операндов.

Условие внутреннего соединения отношений $r_1(S_1)$ и $r_2(S_2)$ также оставим прежнее: P = (b1 = b2). Тогда результатом операции **левого внешнего соединения** будет следующая таблица:

$r_1(S_1) \rightarrow\times_P r_2(S_2)$:

a1	b1	b2	c2
a	1	1	x
b	1	1	x
c	3	3	z
d	4	Null	Null

Действительно, как мы можем видеть, в результате воздействия операции левого внешнего соединения, произошло пополнение результата операции внутреннего соединения несоединимыми кортежами левого, т. е. в нашем случае первого отношения-операнда. Пополнение кортежа на схеме второго (правого) исходного отношения-операнда по определению произошло при помощи Null-значений.

И аналогично результатом **правого внешнего соединения** по тому же, что и раньше, условию P = (b1 = b2) исходных отношений-операндов $r_1(S_1)$ и $r_2(S_2)$ является следующая таблица:

$r_1(S_1) \leftarrow\times_P r_2(S_2)$:

a1	b1	b2	c2
a	1	1	x
b	1	1	x
c	3	3	z
Null	Null	2	y

Действительно, в этом случае пополнять результат операции внутреннего соединения следует несоединимыми кортежами правого, в нашем случае второго исходного отношения-операнда. Такой кортеж, как не трудно видеть, во втором отношении $r_2(S_2)$ один, а именно {2, y}. Далее действуем по определению операции правого внешнего соединения, дополняем кортеж первого (левого) операнда на схеме первого операнда Null-значениями.

И, наконец, рассмотрим третий вариант приведенных ранее операций соединения.

Операция полного внешнего соединения. Эту операцию вполне можно рассматривать не только как операцию, производную от операций внутреннего соединения, но и как объединение операций левого и правого внешнего соединения.

Операция полного внешнего соединения определяется как результат пополнения того же самого внутреннего соединения (как и в случае определения левого и правого внешних соединений) несоединимыми кортежами одновременно и левого, и правого исходных отношений-операндов. Исходя из этого определения дадим формулярный вид этого определения:

$$r_1(S_1) \leftrightarrow\times_P r_2(S_2) = (r_1 \rightarrow\times_P r_2) \# (r_1 \leftarrow\times_P r_2);$$

У операции полного внешнего соединения также имеется свойство, сходное с аналогичным свойством операций левого и правого внешних соединений. Только за счет изначальной взаимно-обратной природы операции полного внешнего соединения (ведь она была определена как объединение операций левого и правого внешних соединений) для нее выполняется **свойство коммутативности**:

$$r_1(S_1) \leftrightarrow\times_P r_2(S_2) = r_2(S_2) \leftrightarrow\times_P r_1(S_1);$$

И для завершения рассмотрения вариантов операций соединения, рассмотрим пример, иллюстрирующий работу операции полного внешнего соединения. Введем два отношения $r_1(S_1)$ и $r_2(S_2)$ и условие соединения.

Пусть

$r_1(S_1)$

a1	b1
a	1
b	1
c	3
d	4

$r_2(S_2)$:

b2	c2
1	x
2	y
3	z

И пусть условием соединения отношений $r_1(S_1)$ и $r_2(S_2)$ будет: P = (b1 = b2), как и в предыдущих примерах.

Тогда результатом операции полного внешнего соединения отношений $r_1(S_1)$ и $r_2(S_2)$ по условию P = (b1 = b2) будет следующая таблица:

$r_1(S_1) \leftrightarrow\times_P r_2(S_2)$:

a1	b1	b2	c2
a	1	1	x
b	1	1	x
c	3	3	z
d	4	Null	Null
Null	Null	2	y

Итак, мы видим, что операция полного внешнего соединения наглядно оправдала свое определение как объединения результатов операций левого и правого внешних соединений. Результирующее отношение операции внутреннего соединения дополнено одновременно несоединимыми кортежами как левого (первого, $r_1(S_1)$), так и правого (второго, $r_2(S_2)$) исходного отношения-операнда.

5. Производные операции

Итак, мы рассмотрели различные варианты операций соединения, а именно операции внутреннего соединения, левого, правого и полного внешнего соединения, которые являются производными восьми исходных операций реляционной алгебры: унарных операций выборки, проекции, переименования и бинарных операций объединения, пересечения, разности, декартова произведения и естественного соединения. Но и среди этих исходных операций есть свои примеры производных операций.

1. Например, операция **пересечения** двух отношений является производной от операции разности этих же двух отношений. Покажем это.

Операцию пересечения можно выразить следующей формулой:

$$r_1(S) \cap r_2(S) = r_1 \setminus r_1 \setminus r_2$$

или, что дает тот же результат:

$$r_1(S) \cap r_2(S) = r_2 \setminus r_2 \setminus r_1;$$

2. Еще одним примером, производной базовой операции от восьми исходных операций является операция **естественного соединения**. В самом общем виде эта операция является производной от бинарной операции декартового произведения и унарных операций выборки, проекции и переименования атрибутов. Однако, в свою очередь, операция внутреннего соединения является производной операцией от той же операции декартового произведения отношений. Поэтому, чтобы показать, что операция естественного соединения – производная операция, рассмотрим следующий пример.

Сравним приведенные ранее примеры для операций естественного и внутреннего соединений.

Пусть нам даны два отношения $r_1(S_1)$ и $r_2(S_2)$ которые будут выступать в качестве операндов. Они равны:

$r_1(S_1)$:

a1	b1
a	1
b	1
c	3
d	4

$r_2(S_2)$:

b2	c2
1	x
2	y
3	z

Как мы уже получали ранее, результатом операции естественного соединения этих отношений будет являться таблица следующего вида:

$r_3(S_3)$ # $r_1(S_1) \times r_2(S_2)$:

a	1	x
b	1	x
c	3	z

А результатом внутреннего соединения этих же отношений $r_1(S_1)$ и $r_2(S_2)$ по условию P = (b1 = b2) будет следующая таблица:

$r_4(S_4)$ # $r_1(S_1) \times_P r_2(S_2)$:

a1	b1	b2	c2
a	1	1	x
b	1	1	x
c	3	3	z

Сравним эти два результата, получившиеся новые отношения $r_3(S_3)$ и $r_4(S_4)$.

Ясно, что операция естественного соединения выражается через операцию внутреннего соединения, но, что главное, с условием соединения специального вида.

Запишем математическую формулу, описывающую действие операции естественного соединения как производную операции внутреннего соединения.

$$r_1(S_1) \times r_2(S_2) = \{ \rho <\#_1> r_1 \times_E \rho< \#_2>r_2\}[S_1 \# S_2],$$

где E — **условие соединимости** кортежей;

$$E= \#a \ \#S_1 \cap S_2 \ [IsNull\ (b1)\ \&\ IsNull\ (2)\ \#b1 = b2];$$

$$b_1 = \#_1\ (name(a)),\ b_2 = \#_2\ (name(a));$$

Здесь одна из **функций переименования** $\#_1$ является тождественной, а другая функция переименования (а именно – $\#_2$) переименовывает атрибуты, на которых наши схемы пересекаются.

Условие соединимости E для кортежей записывается в общем виде с учетом возможных появлений Null-значений, ведь операция внутреннего соединения (как уже было сказано выше) является производной операцией от операции декартового произведения двух отношений и унарной операции выборки.

6. Выражения реляционной алгебры

Покажем, как можно использовать рассмотренные ранее выражения и операции реляционной алгебры в практической эксплуатации различных баз данных.

Конец ознакомительного фрагмента.