Навчальна дисципліна

БАЗИ ДАНИХ

Лектор - к.т.н., доцент

Баклан Ігор Всеволодович

Site: baklaniv.at.ua

E-mail: iaa@ukr.net

2016-2017

Лекція №4.1-4.2.

Реляційна алгебра.Аномалії

(пояснювальні матеріали)

Реляционная алгебра

Обзор реляционной алгебры Кодда

Основная идея *реляционной алгебры* состоит в том, что коль скоро отношения являются множествами, средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для реляционных баз данных.

Существует много подходов к определению *реляционной алгебры*, которые различаются наборами операций и способами их интерпретации, но, в принципе, являются более или менее равносильными. В данном разделе мы рассмотрим немного расширенный начальный вариант *алгебры*, который был предложен Коддом (будем называть ее " *алгеброй Кодда* "). В этом варианте набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса – теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

- *объединения отношений* ;
- *пересечения отношений* ;
- *взятия разности отношений* ;
- *взятия декартова произведения отношений*.

Специальные реляционные операции включают:

- *ограничение отношения* ;
- *проекцию отношения* ;
- *соединение отношений* ;
- *деление отношений*.

Кроме того, в состав *алгебры* включается *операция присваивания*, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и *операция переименования атрибутов*, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

Общая интерпретация реляционных операций

Если не вдаваться в некоторые тонкости, которые мы рассмотрим в следующих разделах, то почти для всех операций предложенного выше набора имеется очевидная и простая интерпретация.

- При выполнении *операции объединения* (`UNION`) двух отношений с одинаковыми заголовками производится отношение, включающее все кортежи, которые входят хотя бы в одно из отношений-операндов.
- *Операция пересечения* (`INTERSECT`) двух отношений с одинаковыми заголовками производит отношение, включающее все кортежи, которые входят в оба отношения-операнда.
- Отношение, являющееся *разностью* (`MINUS`) двух отношений с одинаковыми заголовками, включает все кортежи, входящие в отношение-первый операнд, такие, что ни один из них не входит в отношение, которое является вторым операндом.
- При выполнении *декартова произведения* (`TIMES`) двух отношений, пересечение заголовков которых пусто, производится отношение, кортежи которого производятся путем объединения кортежей первого и второго операндов.
- Результатом *ограничения* (`WHERE`) отношения по некоторому условию является отношение, включающее кортежи отношения-операнда, удовлетворяющее этому условию.
- При выполнении *проекции* (`PROJECT`) отношения на заданное подмножество множества его атрибутов производится отношение, кортежи которого являются соответствующими подмножествами кортежей отношения-операнда.
- При *соединении* (`JOIN`) двух отношений по некоторому условию образуется результирующее отношение, кортежи которого производятся путем объединения кортежей первого и второго отношений и удовлетворяют этому условию.
- У *операции реляционного деления* (`DIVIDE BY`) два операнда – бинарное и унарное отношения. Результирующее отношение состоит из унарных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) включает множество значений второго операнда.
- *Операция переименования* (`RENAME`) производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены.
- *Операция присваивания* (`:=`) позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

Поскольку результатом любой реляционной операции (кроме *операции присваивания*, которая не вырабатывает значения) является некое отношение, можно образовывать реляционные выражения, в которых вместо отношения-операнда некоторой реляционной операции находится вложенное реляционное выражение. В построении реляционного выражения могут участвовать все реляционные операции, кроме *операции присваивания*. Вычислительная интерпретация реляционного выражения диктуется установленными приоритетами операций:

```
RENAME >= WHERE = PROJECT >= TIMES = JOIN = INTERSECT = DIVIDE BY >= UNION = MINUS
```

В другой форме приоритеты операций показаны на рис. 7.1 . Вычисление выражения производится слева направо с учетом приоритетов операций и скобок.

Операция	Приоритет
RENAME	4
WHERE	3
PROJECT	3
TIMES	2
JOIN	2
INTERSECT	2
DIVIDE BY	2
UNION	1
MINUS	1

Рис. 4.1. Таблица приоритетов операций традиционной реляционной алгебры

Замкнутость реляционной алгебры и операция переименования

Как мы отмечали в предыдущей лекции, каждое значение-отношение характеризуется заголовком (или схемой) и телом (или множеством кортежей). Поэтому, если нам действительно нужна *алгебра*, операции которой замкнуты относительно понятия отношения, то каждая операция должна производить отношение в полном смысле, т. е. оно должно обладать и телом, и заголовком. Только в этом случае можно будет строить вложенные выражения.

Заголовок отношения представляет собой множество пар `<имя-атрибута, имя-домена>`. Если посмотреть на общий обзор реляционных операций, приведенный в предыдущем подразделе, то видно, что домены атрибутов результирующего отношения однозначно определяются доменами отношений-операндов. Однако с именами атрибутов результата не всегда все так просто.

Например, представим себе, что у отношений-операндов операции декартова произведения имеются одноименные атрибуты с одинаковыми доменами. Каким был бы заголовок результирующего отношения? Поскольку это множество, в нем не должны содержаться одинаковые элементы. Но и потерять атрибут в результате недопустимо. А это значит, что в таком случае вообще невозможно корректно выполнить операцию декартова произведения.

Аналогичные проблемы могут возникать и в случаях других двуместных операций. Для разрешения проблем в число операций *реляционной алгебры* вводится *операция переименования*. Ее следует применять в том случае, когда возникает конфликт именования атрибутов в отношениях-операндах одной реляционной операции. Тогда к одному из операндов сначала применяется *операция переименования*, а затем основная операция выполняется уже без всяких проблем. Более строго мы определим *операцию переименования* в следующей лекции, а пока лишь заметим, что результатом этой операции является отношение, совпадающее во всем с отношением-операндом, кроме того, что имя указанного атрибута изменено на заданное имя.

В дальнейшем изложении мы будем предполагать применение *операции переименования* во всех конфликтных ситуациях.

Особенности теоретико-множественных операций реляционной алгебры

Хотя в основе теоретико-множественной части *реляционной алгебры Кодда* лежит классическая теория множеств, соответствующие операции *реляционной алгебры* обладают некоторыми особенностями.

Операции объединения, пересечения, взятия разности. Совместимость по объединению

Начнем с *операции объединения отношений* (все, что будет сказано по поводу *объединения*, верно и для *операций пересечения* и *взятия разности отношений*). Смысл *операции объединения* в *реляционной алгебре* в целом остается теоретико-множественным. Напомним, что в теории множеств:

- результатом объединения двух множеств `A{a}` и `B{b}` является такое множество `C{c}`, что для каждого `c` либо существует такой элемент `a`, принадлежащий множеству `A`, что `c=a`, либо существует такой элемент `b`, принадлежащий множеству `B`, что `c=b` ;
- пересечением множеств `A` и `B` является такое множество `C{c}`, что для любого `c` существуют такие элементы `a`, принадлежащий множеству `A`, и `b`, принадлежащий множеству `B`, что `c=a=b` ;
- разностью множеств `A` и `B` является такое множество `C{c}`, что для любого `c` существует такой элемент `a`, принадлежащий множеству `A`, что `c=a`, и не существует такой элемент `b`, принадлежащий `B`, что `c=b`.

Рис. 4.2. Иллюстрация результатов теоретико-множественных операций

Но если в теории множеств операция объединения осмысленна для любых двух множеств-операндов, то в случае *реляционной алгебры* результатом *операции объединения* должно являться отношение. Если в *реляционной алгебре* допустить возможность теоретико-множественного *объединения* двух произвольных отношений (с разными заголовками), то, конечно, результатом операции будет множество, но множество разнотипных кортежей, т. е. не отношение. Если исходить из требования *замкнутости реляционной алгебры* относительно понятия отношения, то такая *операция объединения* является бессмысленной.

Эти соображения подводят к понятию ***совместимости отношений по объединению***: два отношения *совместимы по объединению* в том и только в том случае, когда обладают одинаковыми заголовками. В развернутой форме это означает, что в заголовках обоих отношений содержится один и тот же набор имен атрибутов, и одноименные атрибуты определены на одном и том же домене (эта развернутая формулировка, вообще говоря, является излишней, но она пригодится нам в следующем абзаце).

Если два отношения *совместимы по объединению*, то при обычном выполнении над ними *операций объединения*, *пересечения* и *взятия разности* результатом операции является отношение с корректно определенным заголовком, совпадающим с заголовком каждого из отношений-операндов. Напомним, что если два отношения "почти" *совместимы по объединению*, т. е. совместимы во всем, кроме имен атрибутов, то до выполнения операции типа *объединения* эти отношения можно сделать полностью *совместимыми по объединению* путем применения *операции переименования*.

Для иллюстрации *операций объединения*, *пересечения* и *взятия разности* предположим, что в базе данных имеются два отношения СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 и СЛУЖАЩИЕ_В_ПРОЕКТЕ_2 с одинаковыми схемами {СЛУ_НОМЕР, СЛУ_ИМЯ, СЛУ_ЗАРП, СЛУ_ОТД_НОМЕР} (имена доменов опущены по причине очевидности). Каждое из отношений содержит данные о служащих, участвующих в соответствующем проекте. На рис. 3.3 показано примерное наполнение каждого из двух отношений (некоторые служащие участвуют в обоих проектах).

СЛУЖАЩИЕ_В_ПРОЕКТЕ_1

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2934	Иванов	22000.00	310
2935	Петров	30000.00	310
2936	Сидоров	18000.00	313
2937	Федоров	20000.00	310
2938	Иванова	22000.00	315

СЛУЖАЩИЕ_В_ПРОЕКТЕ_2

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2934	Иванов	22000.00	310
2935	Петров	30000.00	310
2939	Сидоренко	18000.00	313
2940	Федоренко	20000.00	310
2941	Иваненко	22000.00	315

Рис. 4.3. Примерное наполнение отношений СЛУЖАЩИЕ _В_ПРОЕКТЕ_1 и СЛУЖАЩИЕ _В_ПРОЕКТЕ_2

Тогда выполнение операции СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 UNION СЛУЖАЩИЕ_В_ПРОЕКТЕ_2 позволит получить информацию обо всех служащих, участвующих в обоих проектах. Выполнение операции СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 INTERSECT СЛУЖАЩИЕ_В_ПРОЕКТЕ_2 позволит получить данные о служащих, которые одновременно участвуют в двух проектах. Наконец, операция СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 MINUS СЛУЖАЩИЕ_В_ПРОЕКТЕ_2 выработает отношение, содержащее кортежи служащих, которые участвуют только в первом проекте. Результаты этих операций показаны на рис. 4.4.

СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 UNION **СЛУЖАЩИЕ_В_ПРОЕКТЕ_2**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2934	Иванов	22000.00	310
2935	Петров	30000.00	310
2939	Сидоренко	18000.00	313
2940	Федоренко	20000.00	310
2941	Иваненко	22000.00	315
2936	Сидоров	18000.00	313
2937	Федоров	20000.00	310
2938	Иванова	22000.00	315

СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 INTERSECT **СЛУЖАЩИЕ_В_ПРОЕКТЕ_2**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2934	Иванов	22000.00	310
2935	Петров	30000.00	310

СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 MINUS **СЛУЖАЩИЕ_В_ПРОЕКТЕ_2**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2936	Сидоров	18000.00	313
2937	Федоров	20000.00	310
2938	Иванова	22000.00	315

Рис. 4.4. Результаты выполнения операций UNION, INTERSECT и MINUS

Заметим, что включение в состав операций *реляционной алгебры* трех *операций объединения*, *пересечения* и *взятия разности* является, очевидно, избыточным, поскольку, например, операция пересечения выражается через операцию взятия разности[1)]. Тем не менее Кодд в свое время решил включить все три операции, исходя из интуитивных потребностей далекого от математики потенциального пользователя системы реляционных БД.

Операция расширенного декартова произведения и совместимость отношений относительно этой операции

Другие проблемы связаны с операцией взятия декартова произведения двух отношений. В теории множеств декартово произведение может быть получено для любых двух множеств, и элементами результирующего множества являются пары, составленные из элементов первого и второго множеств. Если говорить более точно, декартовым произведением множеств `A{a}` и `B{b}` является такое множество пар `C{<c1, c2>}`, что для каждого элемента `<c1, c2>` множества `C` существуют такой элемент `a` множества `A`, что `c1=a`, и такой элемент `b` множества `B`, что `c2=b`.

Поскольку отношения являются множествами, для любых двух отношений возможно получение прямого произведения. Но результат не будет отношением! Элементами результата будут не кортежи, а пары кортежей.

Поэтому в *реляционной алгебре* используется специализированная форма операции взятия декартова произведения – *расширенное декартово произведение отношений*. При *взятии расширенного декартова произведения* двух отношений элементом результирующего отношения является кортеж, который представляет собой объединение одного кортежа первого отношения и одного кортежа второго отношения.

Приведем более точное определение ***операции расширенного декартова произведения***. Пусть имеются два отношения $R1\{a_1, a_2, \ldots, a_n\}$ и $R2\{b_1, b_2, \ldots, b_m\}$. Тогда результатом операции `R1 TIMES R2` является отношение $R\{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$, тело которого является множеством кортежей вида $\{r_{a1}, r_{a2}, \ldots, r_{an}, r_{b1}, r_{b2}, \ldots, r_{bm}\}$ таких, что $\{r_{a1}, r_{a2}, \ldots, r_{an}\}$ входит в тело `R1`, а $\{r_{b1}, r_{b2}, \ldots, r_{bm}\}$ входит в тело `R2` .

Но теперь возникает вторая проблема – как получить корректно сформированный заголовок отношения-результата? Поскольку схема результирующего отношения является объединением схем отношений-операндов, то очевидной проблемой может быть именование атрибутов результирующего отношения, если отношения-операнды обладают одноименными атрибутами.

Эти соображения приводят к введению понятия *совместимости по взятию расширенного декартова произведения*. Два отношения ***совместимы по взятию расширенного декартова произведения*** в том и только в том случае, если пересечение множеств имен атрибутов, взятых из их схем отношений, пусто. Любые два отношения всегда могут стать *совместимыми по взятию декартова произведения*, если применить *операцию переименования* к одному из этих отношений.

Для наглядности предположим, что в придачу к введенным ранее отношениям `СЛУЖАЩИЕ_В_ПРОЕКТЕ_1` и `СЛУЖАЩИЕ_В_ПРОЕКТЕ_2` в базе данных содержится еще и отношение `ПРОЕКТЫ` со схемой `{ПРОЕКТ_НАЗВ, ПРОЕКТ_РУК}` (имена доменов снова опущены) и телом, показанным на рис. 7.5. На этом же рисунке показан результат операции `СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 TIMES ПРОЕКТЫ`.

ПРОЕКТЫ

ПРОЕКТ_НАЗВ	ПРОЕКТ_РУК
ПРОЕКТ 1	Иванов
ПРОЕКТ 2	Иваненко

СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 TIMES **ПРОЕКТЫ**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР	ПРОЕКТ_НАЗВ	ПРОЕКТ_РУК
2934	Иванов	22000.00	310	ПРОЕКТ 1	Иванов
2935	Петров	30000.00	310	ПРОЕКТ 1	Иванов
2936	Сидоров	18000.00	313	ПРОЕКТ 1	Иванов
2937	Федоров	20000.00	310	ПРОЕКТ 1	Иванов
2938	Иванова	22000.00	315	ПРОЕКТ 1	Иванов
2934	Иванов	22000.00	310	ПРОЕКТ 2	Иваненко
2935	Петров	30000.00	310	ПРОЕКТ 2	Иваненко
2936	Сидоров	18000.00	313	ПРОЕКТ 2	Иваненко
2937	Федоров	20000.00	310	ПРОЕКТ 2	Иваненко
2938	Иванова	22000.00	315	ПРОЕКТ 2	Иваненко

Рис. 4.5. Отношение ПРОЕКТЫ и результат операции СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 TIMES ПРОЕКТЫ

Следует заметить, что *операция взятия декартова произведения* не является слишком осмысленной на практике. Во-первых, мощность тела ее результата очень велика даже при допустимых мощностях операндов, а, во-вторых, результат операции не более информативен, чем взятые в совокупности операнды. Как будет показано далее, основной смысл включения *операции расширенного декартова произведения* в состав *реляционной алгебры Кодда* состоит в том, что на ее основе определяется действительно полезная *операция соединения*.

По поводу теоретико-множественных операций *реляционной алгебры* следует еще заметить, что все четыре операции являются ассоциативными. Т. е. если обозначить через `OP` любую из четырех операций, то `(A OP B) OP C = A OP (B OP C)`, и, следовательно, без внесения двусмысленности можно писать `A OP B OP C` (`A`, `B` и `C` – отношения, обладающие свойствами, необходимыми для корректного выполнения соответствующей операции). Все операции, кроме *взятия разности*, являются коммутативными, т. е. `A OP B = B OP A`.

Специальные реляционные операции

В этом разделе мы несколько подробнее рассмотрим специальные реляционные операции *реляционной алгебры*, такие, как *ограничение*, *проекция*, *соединение* и *деление*.

Операция ограничения

Операция ограничения `WHERE` требует наличия двух операндов: ограничиваемого отношения и простого условия ограничения. Простое условие ограничения может иметь:

- вид (`a comp-op b`), где `a` и `b` – имена атрибутов ограничиваемого отношения; атрибуты `a` и `b` должны быть определены на одном и том же домене, для значений базового типа данных которого поддерживается операция сравнения `comp_op`, или на базовых типах данных, над значениями которых можно выполнять эту операцию сравнения;
- или вид (`a comp-op const`), где `a` – имя атрибута ограничиваемого отношения, а `const` – литерально заданная константа; атрибут `a` должен быть определен на домене или базовом типе, для значений которого поддерживается операция сравнения `comp_op`.

Операцией сравнения `comp-op` могут быть " = ", "  ", " > ", "  ", " < ", "  ". Простые условия вычисляются в трехзначной логике (см. разд. "Реляционная модель данных", лекция 2), и в результате выполнения *операции ограничения* производится отношение, заголовок которого совпадает с заголовком отношения-операнда, а в тело входят те кортежи отношения-операнда, для которых значением условия ограничения является `true`. Тем самым, если в некоторых кортежах содержатся неопределенные значения, и по данной причине вычисление простого условия дает значение `unknown`, то эти кортежи не войдут в результирующее отношение.

Для обозначения вызова *операции ограничения* будем использовать конструкцию `A WHERE comp`, где `A` – ограничиваемое отношение, а `comp` – простое условие сравнения. Пусть `comp1` и `comp2` – два простых условия ограничения. Тогда по определению:

- `A WHERE (comp1 AND comp2)` обозначает то же самое, что и `(A WHERE comp1) INTERSECT (A WHERE comp2)` ;
- `A WHERE (comp1 OR comp2)` обозначает то же самое, что и `(A WHERE comp1) UNION (A WHERE comp2)` ;
- `A WHERE NOT comp1` обозначает то же самое, что и `A MINUS (A WHERE comp1)`.

Эти соглашения позволяют задействовать *операции ограничения*, в которых условием ограничения является произвольное булевское выражение, составленное из простых условий с использованием логических связок `AND`, `OR`, `NOT` и скобок.

Результат выполнения операции `СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 WHERE (СЛУ_ЗАРП > 20000.00 AND (СЛУ_ОТД_НОМ = 310 OR СЛУ_ОТД_НОМ = 315))` (получить данные из отношения `СЛУЖАЩИЕ_В_ПРОЕКТЕ_1` о служащих, работающих в отделах 310 и 315 и получающих зарплату, превышающую 20 000.00 руб.) показан на рис.4.6

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
2934	Иванов	22000.00	310
2935	Петров	30000.00	310
2938	Иванова	22000.00	315

Рис. 4.6. Результат операции СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 WHERE (СЛУ_ЗАРП > 20000.00 AND (СЛУ_ОТД_НОМ = 310 OR СЛУ_ОТД_НОМ = 315))

На интуитивном уровне *операцию ограничения* лучше всего представлять как взятие некоторой "горизонтальной" вырезки из отношения-операнда (выборки некоторых строк из таблицы).

Операция взятия проекции

Операция взятия проекции также требует наличия двух операндов – проецируемого отношения A и подмножества множества имен атрибутов, входящих в заголовок отношения A.

Результатом *проекции отношения* A на множество атрибутов $\{a_1, a_2, \ldots, a_n\}$(PROJECT A $\{a_1, a_2, \ldots, a_n\}$) является отношение с заголовком, определяемым множеством атрибутов $\{a_1, a_2, \ldots, a_n\}$, и с телом, состоящим из кортежей вида $<a_1:v_1, a_2:v_2, \ldots, a_n:v_n>$ таких, что в отношении A имеется кортеж, атрибут a_1 которого имеет значение v_1, атрибут a_2 имеет значение v_2, ..., атрибут a_n имеет значение v_n. Тем самым, при выполнении *операции проекции* выделяется "вертикальная" вырезка отношения-операнда с естественным уничтожением потенциально возникающих кортежей-дубликатов.

Заметим, что потенциальная потребность удаления дубликатов очень сильно усложняет реализацию *операции проекции*, поскольку в общем случае для удаления дубликатов требуется сортировка промежуточного результата операции. Основная сложность состоит в том, что этот промежуточный результат в общем случае может быть очень большим, и для сортировки требуется применять дорогостоящие алгоритмы **внешней сортировки**, выполняемые с применением обменов с внешней памятью. (Под "стоимостью" действия понимается время его выполнения.)

Результат операции PROJECT СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 {СЛУ_ОТД_НОМ} (в каких отделах работают служащие, данные о которых содержатся в отношении СЛУЖАЩИЕ_В_ПРОЕКТЕ_1?) показан на рис. 4.7.

СЛУ_ОТД_НОМЕР
310
313
315

Рис. 4.7. Результат выполнения операции PROJECT СЛУЖАЩИЕ_В_ПРОЕКТЕ_1 {СЛУ_ОТД_НОМ}

Операция соединения отношений

Общая *операция соединения* (называемая также *соединением* по условию) требует наличия двух операндов – соединяемых отношений и третьего операнда – простого условия. Пусть соединяются отношения `A` и `B`. Как и в случае *операции ограничения*, условие соединения `comp` имеет вид либо (`a comp-op b`), либо (`a comp-op const`), где `a` и `b` – имена атрибутов отношений `A` и `B`, `const` – литерально заданная константа, и `comp-op` – допустимая в данном контексте операция сравнения.

Тогда по определению результатом *операции соединения* `A JOIN B WHERE comp` *совместимых по взятию расширенного декартова произведения отношений* `A` и `B` является отношение, получаемое путем выполнения *операции ограничения* по условию `comp` расширенного декартова произведения отношений `A` и `B` $(A\ JOIN\ B\ WHERE\ comp \equiv (A\ TIMES\ B) WHERE\ comp)$.

Если тщательно осмыслить это определение, то станет ясно, что в общем случае применение условия соединения существенно уменьшит мощность результата промежуточного декартова произведения отношений-операндов только в том случае, если условие соединения имеет вид (`a comp-op b`), где `a` и `b` – имена атрибутов разных отношений-операндов. Поэтому на практике обычно считают реальными *операциями соединения* именно те операции, которые основываются на условии соединения приведенного вида.

В подразделе, касающемся *операции ограничения*, мы определили трактовку использования в качестве ограничивающего условия произвольного булевского выражения, которое составлено из простых условий над атрибутами отношения-операнда и литеральными константами. Конечно же, и в *операции соединения* может задаваться произвольное логическое выражение, составленное из простых условий над атрибутами отношений-операндов и константами. *Операцию соединения* с таким условием `comp` разумно считать операцией **действительного** *соединения*, если оно имеет вид (или может быть преобразовано к виду) `comp1 AND (a comp-op b)`, где `a` и `b` – имена атрибутов разных отношений-операндов.

Для иллюстрации *операций соединения* мы немного изменим заголовки и тела отношений, которые использовались ранее в примерах этой лекции. Пусть теперь имеются отношения СЛУЖАЩИЕ {СЛУ_НОМЕР, СЛУ_ИМЯ, СЛУ_ЗАРП, ПРО_НОМ} (атрибут ПРО_НОМ содержит номера проектов, в которых участвует каждый служащий) и ПРОЕКТЫ {ПРО_НОМ, ПРОЕКТ_РУК, ПРО_ЗАРП} (ПРО_НОМ – номер проекта, ПРОЕКТ_РУК – имя служащего-руководителя проекта, ПРО_ЗАРП – средняя заработная плата служащих, участвующих в проекте). Примерное содержимое тел отношений СЛУЖАЩИЕ и ПРОЕКТЫ показано на рис. 7.8.

Тогда осмысленной *операцией соединения* общего вида будет СЛУЖАЩИЕ JOIN ПРОЕКТЫ WHERE (СЛУ_ЗАРП > ПРО_ЗАРП) (выдать данные о служащих, получающих заработную плату, превышающую среднюю заработную плату любого проекта). Результаты этого запроса показаны на рис. 7.9.

Хотя *операция соединения* в приведенной интерпретации не является примитивной (поскольку определяется с использованием *операций декартова произведения* и *проекции*), в силу особой практической важности она включается в базовый набор операций *реляционной алгебры Кодда*. Заметим также, что в практических реализациях *соединение* обычно не выполняется именно как ограничение декартова произведения. Имеются более эффективные алгоритмы, гарантирующие получение такого же результата.

Существует важный частный случай *соединения – эквисоединение* (EQUIJOIN) и простое, но важное расширение операции *эквисоединения – естественное соединение* (NATURAL JOIN). Операция соединения называется ***операцией эквисоединения***, если условие соединения имеет вид (a = b), где a и b – атрибуты разных операндов соединения. Этот случай важен потому, что он чаще всего встречается на практике, и для него существуют наиболее эффективные алгоритмы реализации.

СЛУЖАЩИЕ

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	ПРО_НОМ
2934	Иванов	22400.00	1
2935	Петров	29600.00	1
2936	Сидоров	18000.00	1
2937	Федоров	20000.00	1
2938	Иванова	22000.00	1
2934	Иванов	22400.00	2
2935	Петров	29600.00	2
2939	Сидоренко	18000.00	2
2940	Федоренко	20000.00	2
2941	Иваненко	22000.00	2

ПРОЕКТЫ

ПРО_НОМ	ПРОЕКТ_РУК	ПРО_ЗАРП
1	Иванов	22400.00
2	Иваненко	22400.00

Рис. 4.8. Отношения СЛУЖАЩИЕ и ПРОЕКТЫ

Операция *естественного соединения* применяется к паре отношений `A` и `B`, обладающих (возможно, составным) общим атрибутом `c` (т. е. атрибутом с одним и тем же именем и определенным на одном и том же домене). Пусть `ab` обозначает объединение заголовков отношений `A` и `B`. Тогда ***естественное соединение*** `A` и `B` – это спроецированный на `ab` результат *эквисоединения* `A` и `B` по условию A.c = B.c[2] . Хотя операция *естественного соединения* выражается через *операции переименования*, *соединения* общего вида и *проекции*, для нее обычно используется сокращенная форма, называемая `NATURAL JOIN`.

На рис. 4.10 приведены результаты операций `СЛУЖАЩИЕ JOIN (ПРОЕКТЫ RENAME (ПРО_НОМ, ПРО_НОМ1)) WHERE (СЛУ_ЗАРП = ПРО_ЗАРП)` (*эквисоединение* отношений `СЛУЖАЩИЕ` и `ПРОЕКТЫ`: найти всех служащих, получающих зарплату, равную средней заработной плате в каком-либо проекте) и `СЛУЖАЩИЕ NATURAL JOIN ПРОЕКТЫ` (*естественное соединение* – выдать полную информацию о служащих и проектах, в которых они участвуют).

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	ПРО_НОМ	ПРО_НОМ1	ПРОЕКТ_РУК	ПРО_ЗАРП
2935	Петров	29600.00	1	1	Иванов	22400.00
2935	Петров	29600.00	2	2	Иваненко	22400.00

Рис. 4.9. Результат операции СЛУЖАЩИЕ JOIN ПРОЕКТЫ WHERE (СЛУ_ЗАРП > ПРО_ЗАРП)

Результат операции **СЛУЖАЩИЕ** JOIN (**ПРОЕКТЫ** RENAME (ПРО_НОМ, ПРО_НОМ1)) WHERE (СЛУ_ЗАРП = ПРО_ЗАРП)

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	ПРО_НОМ	ПРО_НОМ1	ПРОЕКТ_РУК	ПРО_ЗАРП
2934	Иванов	22400.00	1	1	Иванов	22400.00
2934	Иванов	22400.00	2	2	Иваненко	22400.00

Результат операции **СЛУЖАЩИЕ** NATURAL JOIN **ПРОЕКТЫ**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	ПРО_НОМ	ПРОЕКТ_РУК	ПРО_ЗАРП
2934	Иванов	22400.00	1	Иванов	22400.00
2935	Петров	29600.00	1	Иванов	22400.00
2936	Сидоров	18000.00	1	Иванов	22400.00
2937	Федоров	20000.00	1	Иванов	22400.00
2938	Иванова	22000.00	1	Иванов	22400.00
2934	Иванов	22400.00	2	Иваненко	22400.00
2935	Петров	29600.00	2	Иваненко	22400.00
2939	Сидоренко	18000.00	2	Иваненко	22400.00
2940	Федоренко	20000.00	2	Иваненко	22400.00
2941	Иваненко	22000.00	2	Иваненко	22400.00

Рис. 4.10. Результаты операций эквисоединения и естественного соединения отношений СЛУЖАЩИЕ и ПРОЕКТЫ

Если вспомнить введенное нами в конце предыдущей лекции определение внешнего ключа отношения, то должно стать понятно, что основной смысл операции *естественного соединения* состоит в возможности восстановления сложной сущности, декомпозированной по причине требования первой нормальной формы. Операция *естественного соединения* не включается в состав набора операций данной *реляционной алгебры Кодда*, но имеет очень важное практическое значение.

Операция деления отношений

Эта операция наименее очевидна из всех операций *реляционной алгебры Кодда* и поэтому нуждается в более подробном объяснении. Пусть заданы два отношения – A с заголовком $\{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$ и B с заголовком $\{b_1, b_2, \ldots, b_m\}$. Будем считать, что атрибут b_i отношения A и атрибут b_i отношения B (i = 1, 2, …, m) не только обладают одним и тем же именем, но и определены на одном и том же домене. Назовем множество атрибутов $\{a_j\}$ составным атрибутом a, а множество атрибутов $\{b_j\}$ – составным атрибутом b. После этого будем говорить о реляционном *делении* "бинарного" отношения A{a, b} на унарное отношение B{b}.

По определению, результатом *деления* A на B (A DIVIDE BY B) является "унарное" отношение C{a}, тело которого состоит из кортежей v таких, что в теле отношения A содержатся кортежи v UNION w такие, что множество {w} включает тело отношения B. *Операция реляционного деления* не является примитивной и выражается через *операции декартова произведения*, *взятия разности* и *проекции*. Мы покажем это в следующей лекции.

Для иллюстрации этой операции предположим, что в базе данных служащих поддерживаются следующие отношения: СЛУЖАЩИЕ, как оно было определено ранее, и унарное отношение НОМЕРА_ПРОЕКТОВ {ПРО_НОМ} (рис. 4.11). Тогда запрос СЛУЖАЩИЕ DIVIDE BY НОМЕРА_ПРОЕКТОВ выдаст данные обо всех служащих, участвующих во всех проектах (результат операции приведен также на рис. 4.11).

Отношение **НОМЕРА_ПРОЕКТОВ**

ПРО_НОМ
1
2

Результат операции **СЛУЖАЩИЕ** DIVIDE BY **НОМЕРА_ПРОЕКТОВ**

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП
2934	Иванов	22400.00
2935	Петров	29600.00

Рис. 4.11. Пример реляционного деления

Заключение

Прежде всего, заметим, что *алгебра Кодда* была представлена не в ее оригинальной форме, а с некоторыми существенными коррективами, внесенными Кристофером Дейтом. Одной из наиболее значительных корректив было добавление тривиальной на первый взгляд *операции переименования атрибутов*. Когда Эдгар Кодд в конце 1960-х гг. впервые опубликовал свою *алгебру*, основное внимание в ней уделялось тому, как конструируются результирующие множества кортежей, т. е. что представляют собой тела результатов операций. Гораздо меньше внимания уделялось заголовкам отношений-результатов. Фактически Кодд пытался применить для именования атрибутов результатов операций точечную нотацию, используя для уточнения имен атрибутов имена исходных отношений-операндов. При наличии произвольно сложных и длинных алгебраических выражений этот путь, в лучшем случае, вел к порождению длинных и трудных для восприятия имен. Очевидно, что введение *операции переименования атрибутов* позволяет легко справиться с этой проблемой.

Далее, *алгебра Кодда* исключительно избыточна. *Операции пересечения*, *декартова произведения* и *естественного соединения*, на самом деле, являются частными случаями одной более общей операции.

Почему же мы начали обсуждение базовых манипуляционных механизмов реляционной модели данных с этой небезупречной и несколько устаревшей *алгебры*? Конечно, прежде всего, из уважения к заслугам доктора Эдгара Кодда, вклад которого в современную технологию баз данных невозможно переоценить. Более практические соображения, повлиявшие на наше решение начать обсуждение с *алгебры Кодда*, заключались в том, что семантика языка SQL во многом базируется именно на этой *алгебре*, и нам будет проще изучать SQL, предварительно познакомившись с ней.

Избыточное дублирование данных и аномалии

Следует различать простое (неизбыточное) и избыточное дублирование данных. Наличие первого из них допускается в базах данных, а избыточное дублирование данных может приводить к проблемам при обработке данных. Приведем примеры обоих вариантов дублирования.

Пример неизбыточного дублирования данных представляет приведенное на рис. 8.1 отношение С_Т с атрибутами Сотрудник и Телефон. Для сотрудников, находящихся в одном помещении, номера телефонов совпадают. Номер телефона 4328 встречается несколько раз, хотя для каждого служащего номер телефона уникален. Поэтому ни один из номеров не является избыточным. Действительно, при удалении одного из номеров телефонов будет утеряна информация о том, по какому номеру можно дозвониться до одного из служащих.

C_T

Сотрудник	Телефон
Иванов	3721
Петров	4328
Сидоров	4328
Егоров	4328

Рис. 8.1. Неизбыточное дублирование

Пример избыточного дублирования (избыточности) представляет приведенное на рис. 8.2 а отношение С_Т_Н, которое, в отличие от отношения С_Т, дополнено атрибутом Н_комн (номер комнаты сотрудника). Естественно предположить, что все служащие в одной комнате имеют один и тот же телефон. Следовательно, в рассматриваемом отношении имеется избыточное дублирование данных. Так, в связи с тем, что Сидоров и Егоров находятся в той же комнате, что и Петров, их номера можно узнать из кортежа со сведениями о Петрове.

С_Т_Н
а)

Сотрудник	Телефон	Н_комн
Иванов	3721	109
Петров	4328	111
Сидоров	4328	111
Егоров	4328	111

С_Т_Н
б)

Сотрудник	Телефон	Н_комн
Иванов	3721	109
Петров	4328	111
Сидоров	---	111
Егоров	---	111

Рис. 8.2. Избыточное дублирование

На рис. 8.2 б приведен пример неудачного отношения С_Т_Н, в котором вместо телефонов Сидорова и Егорова поставлены прочерки (неопределенные значения). Неудачность подобного способа исключения избыточности заключается в следующем. Во-первых, при программировании придется потратить дополнительные усилия на создание механизма поиска информации для прочерков таблицы. Во-вторых, память все равно выделяется под атрибуты с прочерками, хотя дублирование данных и исключено. В-третьих, что особенно важно, при исключении из коллектива Петрова кортеж со сведениями о нем будет исключен из отношения, а значит уничтожена информация о телефоне 111-й комнаты, что недопустимо.

Возможный способ выхода из данной ситуации приведен на рис. 8.3. Здесь показаны два отношения С_Н и Н_Т, полученные путем декомпозиции исходного отношения С_Т_Н. Первое из них содержит информацию о номерах комнат, в которых располагаются сотрудники, а второе - информацию о номерах телефонов в каждой из комнат. Теперь, если Петрова и уволят из учреждения и, как следствие этого, удалят всякую информацию о нем из баз данных учреждения, это не приведет к утере информации о номере телефона в 111-й комнате.

Т_Н

Телефон	Н_комн
3721	109
4328	111

С_Н

Сотрудник	Н_комн
Иванов	109
Петров	111
Сидоров	111
Егоров	111

Рис. 8.3. Исключение избыточного дублирования

Процедура декомпозиции отношения С_Т_Н на два отношения С_Н и Н_Т является основной процедурой нормализации отношений.

Избыточное дублирование данных создает проблемы при обработке кортежей отношения, названные Э. Коддом "аномалиями обновления отношения". Он показал, что для некоторых отношений проблемы возникают при попытке удаления, добавления или редактирования их кортежей.

Аномалиями будем называть такую ситуацию в таблицах БД, которая приводит к противоречиям в БД, либо существенно усложняет обработку данных.

Выделяют три основные вида аномалий: аномалии модификации (или редактирования), аномалии удаления и аномалии добавления.

Аномалии модификации проявляются в том, что изменение значения одного данного может повлечь за собой просмотр всей таблицы и соответствующее изменение некоторых других записей таблицы.

Так, например, изменение номера телефона в комнате 111 (рис. 8.2а), что представляет собой один единственный факт, потребует просмотра всей таблицы С_Т_Н и изменения поля Н_комн согласно текущему содержимому таблицы в записях, относящихся к Петрову, Сидорову и Егорову.

Аномалии удаления состоят в том, что при удалении какого-либо данного из таблицы может пропасть и другая информация, которая не связана напрямую с удаляемым данным.

В той же таблице С_Т_Н удаление записи о сотруднике Иванове (например, по причине увольнения или ухода на заслуженный отдых) приводит к исчезновению информации о номере телефона, установленного в 109-й комнате.

Аномалии добавления возникают в случаях, когда информацию в таблицу нельзя поместить до тех пор, пока она неполная, либо вставка новой записи требует дополнительного просмотра таблицы.

Примером может служить операция добавления нового сотрудника все в ту же таблицу С_Т_Н. Очевидно, будет противоестественным хранение сведений в этой таблице только о комнате и номере телефона в ней, пока никто из сотрудников не помещен в нее. Более того, если в таблице С_Т_Н поле Служащий является ключевым, то хранение в ней неполных записей с отсутствующей фамилией служащего просто недопустимо из-за неопределенности значения ключевого поля.

Вторым примером возникновения аномалии добавления может быть ситуация включения в таблицу нового сотрудника. При добавлении таких записей для исключения противоречий желательно проверить номер телефона и соответствующий номер комнаты хотя бы с одним из сотрудников, сидящих с новым сотрудником в той же комнате. Если же окажется, что у нескольких сотрудников, сидящих в одной комнате, имеются разные телефоны, то вообще не ясно, что делать (то ли в комнате несколько телефонов, то ли какой-то из номеров ошибочный).

Формирование исходного отношения

Проектирование БД начинается с определения всех объектов, сведения о которых будут включены в базу, и определения их атрибутов. Затем атрибуты сводятся в одну таблицу - исходное отношение.

Пример. Формирование исходного отношения.

Предположим, что для учебной части факультета создается БД о преподавателях. На первом этапе проектирования БД в результате общения с заказчиком (заведующим учебной частью) должны быть определены содержащиеся в базе сведения о том, как она должна использоваться и какую информацию заказчик хочет получать в процессе ее эксплуатации. В результате устанавливаются атрибуты, которые должны содержаться в отношениях БД, и связи между ними.

Перечислим имена выделенных атрибутов и их краткие характеристики:

ФИО - фамилия и инициалы преподавателя. Исключаем возможность совпадения фамилии и инициалов у преподавателей.

Должн - должность, занимаемая преподавателем.

Оклад - оклад преподавателя.

Стаж - преподавательский стаж.

Д_Стаж - надбавка за стаж.

Каф - номер кафедры, на которой числится преподаватель.

Предм - название предмета (дисциплины), читаемого преподавателем.

Группа - номер группы, в которой преподаватель проводит занятия.

ВидЗан - вид занятий, проводимых преподавателем в учебной группе.

Одно из требований к отношениям заключается в том, чтобы все атрибуты отношения имели атомарные (простые) значения. В исходном отношении каждый атрибут кортежа также должен быть простым. Пример исходного отношения ПРЕПОДАВАТЕЛЬ приведен на рис. 8.1.1.

ПРЕПОДАВАТЕЛЬ

ФИО	Должн	Оклад	Стаж	ДСтаж	Каф	Предм	Группа	ВидЗан
Иванов И.М.	преп	500	5	100	25	СУБД	256	Практ
Иванов И.М.	преп	500	5	100	25	ПЛ/1	123	Практ
Петров М.И.	ст. преп	800	7	100	25	СУБД	256	Лекция
Петров М.И.	ст. преп	800	7	100	25	Паскаль	256	Практ
Сидоров Н.Г.	преп	500	10	150	25	ПЛ/1	123	Лекция
Сидоров Н.Г.	преп	500	10	150	25	Паскаль	256	Лекция
Егоров В.В.	преп	500	5	100	24	ПЭВМ	244	Лекция

Рис. 8.1.1. Исходное отношение ПРЕПОДАВАТЕЛЬ

Указанное отношение имеет следующую схему ПРЕПОДАВАТЕЛЬ(ФИО, Должн, Оклад, Стаж, Д_Стаж, Каф, Предм, Группа, ВидЗан).

Исходное отношение ПРЕПОДАВАТЕЛЬ содержит избыточное дублирование данных, которое и является причиной аномалий редактирования. Различают избыточность явную и неявную.

Явная избыточность заключается в том, что в отношении ПРЕПОДАВАТЕЛЬ строки с данными о преподавателях, проводящих занятия в нескольких группах, повторяются соответствующее число раз. Например, в отношении ПРЕПОДАВАТЕЛЬ все данные по Иванову повторяются дважды. Поэтому, если Иванов И.М. станет старшим преподавателем, то этот факт должен быть отражен в обеих строках. В противном случае будет иметь место противоречие в данных, что является примером аномалии редактирования, обусловленной явной избыточностью данных в отношении.

Неявная избыточность в отношении ПРЕПОДАВАТЕЛЬ проявляется в одинаковых окладах у всех преподавателей и в одинаковых добавках к окладу за одинаковый стаж. Поэтому, если при изменении окладов за должность с 500 на 510 это значение изменят у всех преподавателей, кроме, например, Сидорова, то база станет противоречивой. Это пример аномалии редактирования для варианта с неявной избыточностью.

Средством исключения избыточности в отношениях и, как следствие, аномалий является нормализация отношений, рассмотрим ее более подробно.

Зависимости между атрибутами

Проектирование БД является одним из этапов жизненного цикла информационной системы. Основной задачей, решаемой в процессе проектирования БД, является задача нормализации ее отношений. Рассматриваемый ниже метод нормальных форм является классическим методом проектирования реляционных БД. Этот метод основан на фундаментальном в теории реляционных баз данных понятии зависимости между атрибутами отношений.

Зависимости между атрибутами

Рассмотрим основные виды зависимостей между атрибутами отношений: функциональные, транзитивные и многозначные.

Понятие функциональной зависимости является базовым, так как на его основе формулируются определения всех остальных видов зависимостей.

Атрибут B **функционально зависит** от атрибута A, если каждому значению A соответствует в точности одно значение B. Математически функциональная зависимость B от A обозначается записью A->B. Это означает, что во всех кортежах с одинаковым значением атрибута A атрибут B будет иметь также одно и то же значение. Отметим, что A и B могут быть составными - состоять из двух и более атрибутов.

В отношении на рис. 5.4 можно выделить функциональные зависимости между атрибутами ФИО->Каф, ФИО->Должн, Должн->Оклад и другие. Наличие функциональной зависимости в отношении определяется природой вещей, информация о которых представлена кортежами отношения. В отношении на рис. 5.4 ключ является составным и состоит из атрибутов ФИО, Предмет, Группа.

Функциональная взаимозависимость. Если существует функциональная зависимость вида A->B и B->A, то между A и B имеется взаимно однозначное соответствие, или функциональная взаимозависимость. Наличие функциональной взаимозависимости между атрибутами A и B обозначим как A<->B или B<->A.

Пример. Пусть имеется некоторое отношение, включающее два атрибута, функционально зависящие друг от друга. Это серия и номер паспорта (N) и фамилия, имя и отчество владельца (ФИО). Наличие функциональной зависимости поля ФИО от N означает не только тот факт, что значение поля N однозначно определяет значение поля ФИО, но и то, что одному и тому же значению поля N соответствует только единственное значение поля ФИО. Понятно, что в данном случае действует и обратная ФЗ: каждому значению поля ФИО соответствует только одно значение поля N. В данном примере предполагается, что ситуация наличия полного совпадения фамилий, имен и отчеств двух людей исключена.

Если отношение находится в 1НФ, то все неключевые атрибуты функционально зависят от ключа с различной степенью зависимости.

Частичной зависимостью (частичной функциональной зависимостью) называется зависимость неключевого атрибута от части составного ключа. В рассматриваемом отношении атрибут Должн находится в функциональной зависимости от атрибута ФИО, являющегося частью ключа. Тем самым атрибут Должн находится в частичной зависимости от ключа отношения.

Альтернативным вариантом является **полная функциональная зависимость** неключевого атрибута от всего составного ключа. В нашем примере атрибут ВидЗан находится в полной функциональной зависимости от составного ключа.

Атрибут C зависит от атрибута A **транзитивно** (существует **транзитивная зависимость**), если для атрибутов A, B, C выполняются условия A->B и B->C, но обратная зависимость отсутствует. В отношении на рис. 5.4 транзитивной зависимостью связаны атрибуты:

ФИО->Должн->Оклад Между атрибутами может иметь место многозначная зависимость.

В отношении R атрибут B **многозначно зависит** от атрибута A, если каждому значению A соответствует множество значений B, не связанных с другими атрибутами из R.

Многозначные зависимости могут быть "один ко многим" (1:M), "многие к одному" (M: 1) или "многие ко многим" (M:M), обозначаемые соответственно: A=>B, A<=B и A<=>B.

Например, пусть преподаватель ведет несколько предметов, а каждый предмет может вестись несколькими преподавателями, тогда имеет место зависимость ФИО<=>Предмет. Так, из таблицы 7.2, приведенной на рис. 5.4., видно, что преподаватель Иванов И.М. ведет занятия по двум предметам, а дисциплина СУБД - читается двумя преподавателями: Ивановым И.М. и Петровым М.И.

Замечание. В общем случае между двумя атрибутами одного отношения могут существовать зависимости: 1:1,1:M,M:1 и M:M. Поскольку зависимость между атрибутами является причиной аномалий, стараются расчленить отношения с зависимостями атрибутов на несколько отношений. В результате образуется совокупность связанных отношений (таблиц) со связями вида 1:1,1:M, M:1 и M:M. Связи между таблицами отражают зависимости между атрибутами различных отношений.

Взаимно независимые атрибуты. Два или более атрибута называются взаимно независимыми, если ни один из этих атрибутов не является функционально зависимым от других атрибутов.

В случае двух атрибутов отсутствие зависимости атрибута A от атрибута B можно обозначить так: A-.->B. Случай, когда A-.->B и B-i->A, можно обозначить A-.=B.

Выявление зависимостей между атрибутами

Выявление зависимостей между атрибутами необходимо для выполнения проектирования БД методом нормальных форм, рассматриваемого далее.

Основной способ определения наличия функциональных зависимостей - внимательный анализ семантики атрибутов. Для каждого отношения существует, но не всегда, определенное множество функциональных зависимостей между атрибутами. Причем если в некотором отношении существует одна или несколько функциональных зависимостей, можно вывести другие функциональные зависимости, существующие в этом отношении.

Пример. Пусть задано отношение R со схемой R(A1, A2, A3) и числовыми значениями, приведенными в следующей таблице:

A1	A2	A3
12	21	34
17	21	34
11	24	33
13	25	31
15	23	35
14	22	32

Априори известно, что в R существуют функциональные зависимости:

A1-> A2 и A2-> A3.

Анализируя это отношение, можно увидеть, что в нем существуют еще зависимости:

A1-> A3, A1A2-> A3, A1A2A3-> A1A2, A1A2-> A2A3 и т. п.

В то же время в отношении нет других функциональных зависимостей, что во введенных нами обозначениях можно отразить следующим образом:

A2-,->A1, A3-,->A1 и т. д.

Отсутствие зависимости A1 от A2 (A2-.->A1) объясняется тем, что одному и тому же значению атрибута A2 (21) соответствуют разные значения атрибута A1 (12 и 17). Другими словами, имеет место многозначность, а не функциональность.

Перечислив все существующие функциональные зависимости в отношении R, получим полное множество функциональных зависимостей, которое обозначим F +.

Таким образом, для последнего примера исходное множество F = (A1-> A2, A2-> A3), а полное множество F += (A1-> A2, A2-> A3, A1-> A3, A1A2-> A3, A1A2A3-> A1A2, A1A2->A2A3,...).

Для построения F+ из F необходимо знать ряд правил (или аксиом) вывода одних функциональных зависимостей из других.

Существует 8 основных аксиом вывода:

рефлексивности
пополнения
транзитивности
расширения
продолжения
псевдотранзитивности
объединения
декомпозиции.

Перечисленные аксиомы обеспечивают получение всех ФЗ, т. е. их совокупность применительно к процедуре вывода можно считать "функционально полной".

Выявим зависимости между атрибутами отношения ПРЕПОДАВАТЕЛЬ, приведенного на рис. 5.4. При этом учтем следующее условие, которое выполняется в данном отношении: один преподаватель в одной группе может проводить один вид занятий (лекции или практические занятия).

В результате анализа отношения получаем зависимости между атрибутами, показанные на рис. 8.2.1.

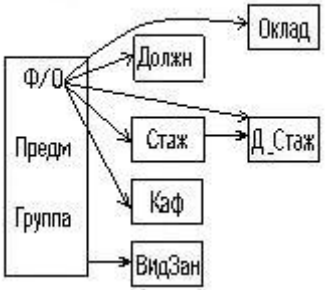

Рис. 8.2.1 Зависимости между атрибутами

ФИО->Оклад
ФИО->Должн
ФИО->Стаж
ФИО->Д_Стаж
ФИО->Каф
Стаж->Д_Стаж
Должн->0клад
Оклад->Должк
ФИО.Предм.Группа->ВидЗан

К выделению этих ФЗ для рассматриваемого примера приводят следующие соображения.

Фамилия, имя и отчество у преподавателей факультета уникальны. Каждому преподавателю однозначно соответствует его стаж, т. е. имеет место функциональная зависимость ФИО->Стаж. Обратное утверждение неверно, так как одинаковый стаж может быть у разных преподавателей.

Каждый преподаватель имеет определенную добавку за стаж, т. е. имеет место функциональная зависимость ФИО->Д_Стаж, но обратная функциональная зависимость отсутствует, так как одну и ту же надбавку могут иметь несколько преподавателей.

Каждый преподаватель имеет определенную должность (преп., ст.преп., доцент, профессор), но одну и ту же должность могут иметь несколько преподавателей, т. е. имеет место функциональная зависимость ФИО->Должн, а обратная функциональная зависимость отсутствует.

Каждый преподаватель является сотрудником одной и только одной кафедры. Поэтому функциональная зависимость ФИО->Каф имеет место. С другой стороны, на каждой кафедре много преподавателей, поэтому обратной функциональной зависимости нет.

Каждому преподавателю соответствует конкретный оклад, который одинаков для всех педагогов с одинаковыми должностями, что учитывается зависимостями ФИО->Оклад и Должн->Оклад. Нет одинаковых окладов для разных должностей, поэтому имеет место функциональная зависимость Оклад->Должн.

Один и тот же преподаватель в одной группе по разным предметам может проводить разные виды занятий. Определение вида занятий, которые проводит преподаватель, невозможно без указания предмета и группы, поэтому имеет место функциональная зависимость ФИО, Предм, Группа->ВидЗан. Действительно, Петров М.И. в 256 группе читает лекции и проводит пратические занятия. Но лекции он читает по СУБД, а практику проводит по Паскалю.

Нами не были выделены зависимости между атрибутами ФИО, Предм и Группа, поскольку они образуют составной ключ и не учитываются в процессе нормализации исходного отношения.

После того, как выделены все функциональные зависимости, следует проверить их согласованность с данными исходного отношения ПРЕПОДАВАТЕЛЬ.

Например, Должн.=преп и Оклад=500 всегда соответствуют друг другу во всех кортежах, т. е. подтверждается функциональная зависимость Должн<->Оклад. Так же следует верифицировать и остальные функциональные зависимости, не забывая об ограниченности имеющихся в отношении данных.