Реляционная алгебра

Реляционная алгебра является основным компонентом реляционной модели, опубликованной Коддом, и состоит из восьми операторов, составляющих две группы по четыре оператора:

- Традиционные операции над множествами: объединение (UNION), пересечение (INTERSECT), разность (MINUS) и декартово произведение (TIMES). Все операции модифицированы, с учетом того, что их операндами являются отношения, а не произвольные множества.
- Специальные реляционные операции: ограничение (WHERE) , проекция (PROJECT), соединение (JOIN) и деление (DIVIDE BY).

Результат выполнения любой операции реляционной алгебры над отношениями также является отношением. Эта особенность называется свойством реляционной замкнутости. Утверждается, что поскольку реляционная алгебра является замкнутой, то в реляционных выражениях можно использовать вложенные выражения сколь угодно сложной структуры. Если рассматривать свойство реляционной замкнутости строго, то каждая реляционная операция должна быть определена таким образом, чтобы выдавать результат с надлежащим типом отношения (в частности, с соответствующим набором атрибутов или заголовком). Для достижения этой цели вводится новый оператор переименование (RENAME), предназначенный для переименования атрибутов в определенном отношении.

В качестве основы для последующих обсуждений рассмотрим упрощенный синтаксис выражений реляционной алгебры в форме БНФ.

```
реляционное_выражение ::= унарное_выражение | бинарное_выражение
унарное_выражение ::= переименование | ограничение | проекция
переименование ::= терм RENAME имя_атрибута AS имя_атрибута
терм ::= имя_отношения | ( реляционное_выражение )
ограничение ::= терм WHERE логическое_выражение
проекция ::= терм | терм [ список_имен_атрибутов ]
бинарное_выражение ::= проекция бинарная_операция реляционное_выражение
бинарная_операция :: = UNION | INTERSECT | MINUS | TIMES | JOIN | DIVIDEBY
```

По приведенной грамматике можно сделать следующие замечания.

- Реляционные операторы UNION, INTERSECT и MINUS требуют, чтобы отношения были совместимыми по типу, т. е. имели идентичные заголовки.
- Реляционные операторы UNION, INTERSECT, TIMES и JOIN ассоциативны и коммутативны.
- Если отношения A и B не имеют общих атрибутов, то операция соединения A JOIN B эквивалентна операции A
  TIMES B, т. е. в таком случае соединение вырождается в декартово произведение. Такое соединение называют естественным.
- Другой допустимый синтаксис для синтаксической категории переименования таков: (терм RENAME список- переименований). Здесь каждый из элементов списка переименований представляет собой выражение имя_атрибута AS имя_атрибута.
- Несмотря на большие возможности, предоставляемые операторами реляционной алгебры, существует несколько типов запросов, которые нельзя выразить этими средствами. Для таких случаев необходимо использовать процедурные расширения реляционных языков.

В алгебре Кодда не все операторы являются независимыми, т. е. некоторые из реляционных операторов могут быть выражены через другие реляционные операторы.

Оператор естественного соединения по атрибуту Y определяется через оператор декартового произведения и оператор ограничения:

A JOIN B = ((A TIMES (B RENAME Y AS Y1)) WHERE Y=Y1)[X, Y, Z]

Оператор пересечения выражается через вычитание следующим образом:

A INTERSECT B = A MINUS (A MINUS B)

Оператор деления выражается через операторы вычитания, декартового произведения и проекции следующим образом:

A DIVIDEBY B = A[X] MINUS ((A[X] TIMES B) MINUS A)[X]

Оставшиеся реляционные операторы (объединение, вычитание, декартово произведение, ограничение, проекция) являются примитивными операторами – их нельзя выразить друг через друга.

В качестве примера рассмотрим запросы на языке реляционной алгебры для схемы базы данных «Поставщики и детали», представленной следующими схемами отношений:

S(Sno: integer, Sname: string, Status: integer, City: string)
P(Pno: integer, Pname: string, Color: string, Weight: real, City: string)
SP(Sno: integer, Pno: integer, Qty: integer)

В данном примере имена доменов представлены именами типов, имена типов отделяются от имен атрибутов двоеточием, первичные ключи выделены подчеркиванием, а имена внешних ключей схемы отношения SP (ПОСТАВКА) совпадают с именами первичных ключей схем отношений S (ПОСТАВЩИК) и P (ДЕТАЛЬ).

Получить имена поставщиков, которые поставляют деталь под номером 2	((SP JOIN S) WHEPE Pno = 2) [Sname]
Получить имена поставщиков, которые поставляют по крайней мере одну красную деталь	(((P WHERE Color = 'Красный') JOIN SP) [Sno] JOIN S) [Sname] Другая формулировка того же запроса: (((P WHERE Color = 'Красный') [Pno] JOIN SP) JOIN S) [Sname] Этот пример подчеркивает одно важное обстоятельство: возможность сформулировать один и тот же запрос несколькими способами.
Получить имена поставщиков, которые поставляют все детали	((SP [Sno, Pno] DIVIDE BY P [Pno] JOIN S) [Sname]

Получить номера поставщиков, поставляющих по крайней мере все те детали, которые поставляет поставщик под номером 2	SP [Sno, Pno] DIVIDE BY (SP WHEPE Sno = 2) [Pno]
Получить все пары номеров поставщиков, размещенных в одном городе	(((S RENAME Sno AS FirstSno) [FirstSno, City] JOIN (S RENAME Sno AS SecondSno) [SecondSno , City]) WHEPE FirstSno < SecondSno) [FirstSno, SecondSno]
Получить имена поставщиков, которые не поставляют деталь под номером 2	((S[Sno] MINUS (SP WHEPE Pno = 2) [Sno]) JOIN S) [Sname]

Вычислительные возможности реляционной алгебры можно увеличить путем введения дополнительных операторов. Дополнительные операторы реляционной алгебры

Многие авторы предлагали новые алгебраические операторы после определения Коддом первоначальных восьми. Рассмотрим несколько таких операторов:

SEMIJOIN (полусоединение), SEMIMINUS (полувычитание), EXTEND (расширение), SUMMARIZE (обобщение) и TCLOSE (транзитивное замыкание).

Синтаксис этих операторов выглядит следующим образом:

<полусоединение> ::= <реляционное выражение> SEMIJOIN <реляционное выражение>
< полувычитание > ::= <реляционное выражение> SEMIMINUS <реляционное выражение>
< расширение> ::= EXTEND <реляционное выражение> ADD
<список добавляемых расширений>
<добавляемое расширение> ::= <выражение> AS <имя атрибута>
<обобщение> ::= SUMMARIZE <реляционное выражение> PER <реляционное выражение> ADD <список добавляемых обобщений>
<добавляемое обобщение> ::= <тип обобщения> [(<скалярное выражение>)] AS <имя атрибута> <тип обобщения> ::= COUNT | SUM | AVG | MAX | MIN | ALL | ANY | COUNTD | SUMD | AVGD <транзитивное замыкание> ::= TCLOSE <реляционное выражение>

Операция расширения

С помощью операции расширения из определенного отношения (по крайней мере, концептуально) создается новое отношение, которое содержит дополнительный атрибут, значения которого получены посредством некоторых скалярных вычислений.

EXTEND S ADD 'Supplier' AS TAG
EXTEND P ADD (Weight * 454) AS GMWT
(EXTEND P ADD (Weight * 454) AS GMWT) WHERE GMWT > Weight (10000.0)) { ALL BUT GMWT } EXTEND (P JOIN SP) ADD (Weight * Qty) AS SHIPWT
(EXTEND S ADD City AS SCity) { ALL BUT City }
EXTEND P ADD Weight * 454 AS GMW, Weight * 16 AS OZWT)
EXTEND S ADD COUNT ((SP RENAME SNo AS X) WHERE X = SNo) AS NP

Рассмотрим вкратце обобщающие функции. Общее назначение этих функций состоит в том, чтобы на основе значений некоторого атрибута определенного отношения получить скалярное значение. В языке Tutorial D параметр <вызов обобщающей функции> является особым случаем параметра <скалярное выражение> и в общем случае имеет следующий вид.

<имя функции> (<реляционное выражение> [, <имя атрибута>])
Если параметр <имя функции> имеет значение COUNT, то параметр <имя атрибута> недопустим и должен быть опущен. В остальных случаях параметр <имя атрибута> может быть опущен тогда и только тогда, когда параметр <реляционное выражение> задает отношение со степенью, равной единице.

Операция обобщения

В реляционной алгебре операция расширения позволяет выполнять "горизонтальные" вычисления в отношении отдельных строк. Оператор обобщения выполняет аналогичную функцию для "вертикальных" вычислений в отношении отдельного столбца. Примеры.

SUMMARIZE SP PER SP { PNo } ADD SUM (Qty) AS TOTQTY

В результате его вычисления создается отношение с заголовком {P#, TOTQT Y}, содержащее один кортеж для каждого значения атрибута P# в проекции SP{P#}. Каждый из этих кортежей содержит значение атрибута P# и соответствующее общее количество деталей. Другими словами, концептуально исходное отношение P «перегруппировано» в множество групп кортежей (по одной группе для каждого уникального значения атрибута P#), после чего для каждой полученной группы сгенерирован один кортеж, помещаемый в окончательный результат.

В общем случае выражение выглядит: SUMMARIZE A PER B ADD <обобщение> AS

Определяется следующим образом:

- Отношение B должно иметь такой же тип, как и некоторая проекция отношения A, T.e. каждый атрибут отношения B должен одновременно присутствовать в отношении A. Примем, что атрибутами этой проекции (или, что эквивалентно, атрибутами отношения B) являются атрибуты A1, A2, ... , An.
- Результатом вычисления данного выражения будет отношение с заголовком {Al, A2, ... , An, Z}, где Z является новым добавленным атрибутом.
- Тело результата содержит все кортежи t, где t является кортежем отношения B, расширенным значением нового атрибута Z. Это значение нового атрибута Z подсчитывается посредством вычисления обобщающего выражения по всем кортежам отношения A, которое имеет те же значения для атрибутов A1, A2, ... , An, что и кортеж t. (Разумеется, если в отношении A нет кортежей, принимающих те же значения для атрибутов A1, A2, .. . , An, что и кортеж t, то обобщающее выражение будет вычислено для пустого множества.) Отношение B не должно со- держать атрибут с именем Z, а обобщающее выражение не должно ссылаться на атрибут Z. Заметьте, что кардинальность результата равна кардинальности отношения B, а степень результата равна степени отношения B плюс единица. Типом переменной Z в этом случае будет тип обобщающего выражения.

Вот еще один пример:
SUMMARIZE (P JOIN SP) PER P { City } ADD COUNT AS NSP

Легко заметить, что оператор SUMMARIZE не примитивен – его можно моделировать с помощью оператора EXTEND. Рассмотрим следующее выражение

SUMMARIZE SP PER S { SNo } ADD COUNT AS NP
По сути, это сокращенная запись представленного ниже более сложного выражения
{ EXTEND S { Sno } ADD ((SP RENAME Sno AS X) WHERE X=Sno) AS Y, COUNT (Y) AS NP) { Sno, NP }

Группирование и разгруппирование

Поскольку значениями атрибутов отношений могут быть другие отношения, было бы желательным наличие дополнительных реляционных операторов, называемых операторами группирования и разгруппирования. Рассмотрим пример SP GROUP (PNo, Qty) AS PQ который можно прочесть как «сгруппировать отношение SP по атрибуту SNo», поскольку атрибут SNo является единственным атрибутом отношения SP, не упомянутым в предложении GROUP. В результате получится отношение, заголовок которого выглядит так

{ SNo SNo, PQ RELATION { PNo PNo, Qty Qty } }

Другими словами, он состоит из атрибута PQ, принимающего в качестве значений отношения (PQ, в свою очередь, имеет атрибуты PNo и Qty), а также из всех остальных атрибутов отношения SP (в нашем случае "все остальные атрибуты" - это атрибут SNo). Тело этого отношения содержит ровно по одному кортежу для всех различных значений атрибута SNo исходного отношения SP.

Перейдем теперь к операций разгруппирования. Пусть SPQ - это отношение, полученное в результате группирования. Тогда выражение SPQ UNGROUP PQ возвращает нас к отношению SP (как и следовало ожидать). Точнее, оно выдает в качестве результата отношение, заголовок которого выглядит так (SNo SNo, PNo PNo, Qty Qty)

Реляционные сравнения

Реляционная алгебра в том виде, в котором она была изначально определена, не поддерживает прямого сравнения двух отношений (например, проверки их равенства или того, является ли одно из них подмножеством другого). Это упущение легко исправляется следующим образом. Сначала определяется новый вид условия - реляционное сравнение - со следующим синтаксисом.

<реляционное выражение> <отношение> <реляционное выражение> <отношение> ::=
> -- Собственное супермножество | >= -- Супермножество
| < -- Собственное подмножество | <= -- Подмножество
| = -- Равно
| <> -- Не равно
Здесь параметр <реляционное выражение> в обоих случаях выражения реляционной алгебры, представляющие совместимые по типу отношения.

S (City) = P (City)

Смысл выражения: совпадает ли проекция отношения поставщиков S по атрибуту City с проекцией отношения деталей P по атрибуту City? S (SNo) > SPJ (SNo)

Смысл выражения: есть ли поставщики, вообще не поставляющие деталей?
На практике часто требуется определить, является ли данное отношение пустым. Соответствующий оператор, возвращающий логическое значение имеет вид

IS_EMPTY (<реляционное выражение>)
Не менее часто требуется проверить, присутствует ли данный кортеж t в данном отношении R. Для этой цели подойдет следующее реляционное сравнение.

RELATION { t } <= R
Однако, с точки зрения пользователя, удобнее применять следующее сокращение: t IN R