УДК 681.3

В.А. Резниченко
Институт программных систем НАН Украины

Выражение предикатов над множествами в SQL

Рассматриваются возможности SQL по выражению теоретико-множественных предикатов. Возможности языка раскрываются на многочисленных примерах запросов. Для демонстрации запросов используется СУБД Oracle.

Розглядаються можливості SQL по представленню теоретико-множинних предикатів. Можливості мови розкриваються на численних прикладах запитів. Для демонстрації запитів використовується СУБД Oracle.

The possibilities of SQL to express of set-theoretic predicates are considered. The language features are represented in numerous examples of queries. To demonstrate the queries the DBMS Oracle is used.

Ключевые слова: база данных, язык запросов, язык SQL, теоретико-множественные предикаты

1 В чем суть вопроса?

К сожалению, в стандарте SQL нет предикатов типа CONTAINS и IS CONTAINED IN, которые проверяют вхождение множества во множество. С их помощью можно было бы довольно легко выражать запросы, эквивалентные использованию операции деления в реляционной алгебре или квантора общности в реляционном исчислении. Тем не менее, в SQL имеются возможности выражать любые предикаты над множествами.

Давайте рассмотрим на примере, что имеется в виду. Пусть имеются две следующих таблицы:

```
TEACHER (TchPK, Name)
LECTURE (TchFK, Type)
```

где TchPK - первичный ключ таблицы преподавателей, Name - фамилии преподавателей, TchFK – внешний ключ, ссылающийся на преподавателя, Type - тип занятия.

Нам следует получить ответ на следующий запрос:

Запрос. Вывести фамилии преподавателей, множество типов занятий которых включает в себя множество всех имеющихся типов занятий.

В качестве критерия BCEX типов занятий используется множество всех различных значений в столбце Type таблицы LECTURE. Если рассуждать с позиции понятия множеств, то для получения ответа на запрос следует поступить следующим образом (см. рис 1). Прежде всего следует построить множество всех возможных типов занятий (множество B). Затем, для каждого преподавателя строим множество типов занятий, которые у него имеются (множество A). Наконец, если множество A включает в себя множество B, то такой преподаватель попадает в результат выполнения запроса.

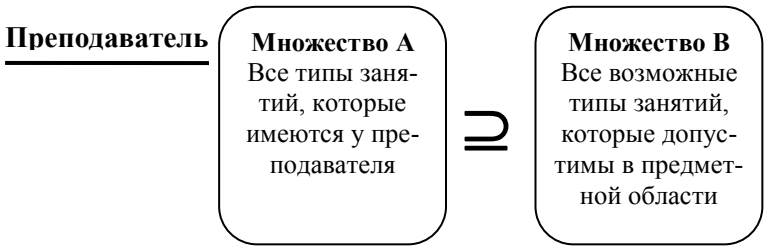


Рис. 1 Множественная интерпретация выполнения запроса

Если предикат включения «⊇» обозначить через CONTAINS и предположить, что он реализован в SQL, то тогда этот запрос мог бы выглядеть следующим образом:

```
SELECT Name
FROM   TEACHER t
WHERE  (SELECT DISTINCT Type FROM LECTURE l
        WHERE l.TchFK = t.TchPK)
         CONTAINS
        (SELECT DISTINCT Type FROM LECTURE);
```

Однако такого предиката в SQL нет. Тем не менее, запросы такого типа выразимы в SQL. Суть их построения заключается в следующем.

Такие запросы можно всегда переформулировать таким образом, чтобы для их интерпретации не надо было бы формировать множества, а только работать со строками таблиц, правда для этого следует использовать так называемые кванторы общности (ДЛЯ ВСЕХ) и существования (СУЩЕСТВУЕТ). Так, например, предыдущий запрос можно переформулировать следующим образом: «Вывести фамилии таких преподавателей, у которых для ВСЕХ типов занятий СУЩЕСТВУЮТ соответствующие этому типу лекции» или, проще говоря «Вывести фамилии преподавателей, имеющих все типы занятий».

К сожалению, для непосредственной записи в SQL таким образом сформулированного запроса необходим квантор общности, который отсутствует в SQL. Тем не менее, запрос можно переформулировать таким образом, что становится понятно, как его можно описать имеющимися средствами SQL. Для этого напомним, что квантор общности выражается через отрицания и квантор существования. А именно высказывание типа «для всех x имеет место A» может быть заменено на следующее: «не существуют такие x, что не имеет место A». В связи с этим наш запрос можно сформулировать так: «Вывести фамилии преподавателей, для которых не существуют такие типы занятий, которые они не проводят». Здесь для непосредственно записи запроса необходимы отрицания и квантор существования, которые в стандарте SQL имеются.

Теперь покажем, как наш запрос записывается в SQL.

Первая часть этого переформулированного запроса ("Выдать преподавателей, для которых не существуют такие типы занятий, которые...") выражается следующим образом:

```
SELECT t.Name
FROM   TEACHER t
WHERE  NOT EXISTS (SELECT l.Type
                   FROM   LECTURE l
                   WHERE  ...);
```

Теперь следует записать выражение, которое определяет условие на тип занятия. Для этого сначала выразим следующий вспомогательный запрос: "Выдать такие типы занятий, которые не читает преподаватель Иванов". Он имеет следующий вид:

```
SELECT l.Type
FROM   LECTURE l
WHERE  NOT EXISTS
       (SELECT l2.TchFK
        FROM   LECTURE l2, TEACHER t2
        WHERE  t2.TchPK = l2.TchFK AND
               t2.Name = 'Иванов' AND
               l2.Type = l.Type);
```

Теперь эти два запроса необходимо скомбинировать следующим образом:

- фраза WHERE второго запроса (внешнего) заменяет фразу WHERE вложенного подзапроса первого запроса;
- во фразе WHERE второго запроса вместо константы конкретного имени преподавателя (Иванов) используется имя преподавателя, определяемое переменной первого запроса (t.name).

В результате получаем необходимый запрос.

```
SELECT t.Name
FROM   TEACHER t
WHERE  NOT EXISTS
          (SELECT l.Type
           FROM   LECTURE l
           WHERE  NOT EXISTS
                     (SELECT l2.TchFK
                      FROM   LECTURE l2, TEACHER t2
                      WHERE  t2.TchPK = l2.TchFK AND
                             t2.Name = t.Name AND
                             l2.Type = l.Type));
```

строки не выбраны

Как видим, таких преподавателей в нашей базе данных не оказалось. У нас было очень сильное условие на типы занятий, а именно, все типы занятий. Давайте ослабим его.

Запрос. Вывести фамилии преподавателей, которые имеют, по крайней мере, те типы занятий, которые имеет преподаватель Хоренко. Или, другими словами, следует вывести фамилии преподавателей, для которых среди типов занятий, проводимых Хоренко не существуют такие, которые они не проводят.

Как и в предыдущем случае, приведем сначала промежуточный запрос: "Среди типов занятий, которые имеет преподаватель Хоренко, выдать такие, которые не имеет преподаватель Иванов", Он имеет следующий вид:

```
SELECT l.Type
FROM   LECTURE l, TEACHER t
WHERE  l.TchFK = t.TchPK AND t.Name = 'Хоренко' AND
       NOT EXISTS (SELECT l2.TchFK
                   FROM   LECTURE l2, TEACHER t2
                   WHERE  t2.TchPK = l2.TchFK AND
                          t2.Name = 'Иванов' AND
                          l2.Type = l.Type);
```

Здесь по сравнению с первым примером мы во фразе `WHERE` внешнего запрос добавили условие, выбирающее типы занятий преподавателя Хоренко.

Теперь, объединяя этот запрос так, как мы поступили в предыдущем примере, получим следующий результирующий запрос.

```
SELECT tch.Name
FROM   TEACHER tch
WHERE  NOT EXISTS
          (SELECT DISTINCT l.Type
           FROM   LECTURE l, TEACHER t
           WHERE  l.TchFK = t.TchPK AND
                  t.Name = 'Хоренко' AND
                  NOT EXISTS
                     (SELECT DISTINCT l2.TchFK
                      FROM   LECTURE l2,TEACHER t2
                      WHERE  t2.TchPK = l2.TchFK AND
                             t2.Name = tch.Name AND
                             l2.Type = l.Type));
```

```
NAME
--------------
Рамишевский
Хоренко
Кузинцев
Завратинский
Лекарь
Резван
Дараганов
7 строк выбрано.
```

Согласно нашей базе данных Хоренко проводит только лекционные занятия, поэтому выбраны те преподаватели, которые проводят, по крайней мере, лекционные занятия. Если в этом запросе заменить преподавателя Хоменко Воропаевым, который проводит только лабораторные занятия, то получим тех преподавателей, которые проводят, по крайней мере, лабораторные занятия и т.д.

Итак, имеется возможность записывать запросы в SQL, которые предполагают проверку вхождения множества во множество, однако, как это видно из наших примеров, это не так-то легко сделать.

Более того, мы рассмотрели только один из теоретико-множественных предикатов и только один частный случай его использования.

В связи с этим в следующих разделах мы введем дополнительный предикат, которого нет в SQL, но который позволяет существенно упростить формулировку подобных запросов, и затем приведем простой алгоритм преобразования запросов с таким предикатом в стандартные возможности SQL. Затем мы рассмотрим, каким образом можно выражать в стандартном SQL все существующие теоретико-множественные предикаты. И, наконец, обсудим некоторые другие аспекты использования этих предикатов.

2 Использование предиката FORALL

Давайте введем в SQL новый предикат FORALL, который позаимствован из статьи [1] Он имеет следующий синтаксис:

```
FORALL (запрос ON условие)
```

Если этот предикат принимает значение true на текущей строке таблицы, то эта строка попадает в результат вычисления запроса. Он же принимает истинное значение, если для всех строк его запроса задаваемое во фразе ON условие оказывается истинным. Давайте обозначим через A выражение, которое формирует запрос, а B – соответствующее условию. Тогда формальный смысл этого предиката следующий:

Для любого x, если истинно A(x), то отсюда следует истинность B(x).

В этой формулировке A называется *посылкой* нашего рассуждения, а B — *заключением*. Обратите внимание, что это не одно и то же, что следующее утверждение:

Для любого x должны быть истинными и A(x) и B(x).

Всякий раз, когда вы анализируете возможность применения предиката FORALL, убедитесь, что ваш содержательный запрос имеет именно первый из перечисленных смыслов.

2.1 Демонстрационная база данных

Во всех последующих примерах мы будем использовать базу данных, имеющую следующую структуру:

- CAR(CRID, Type) — перечень типов автомобилей: ID и тип;
- COMPANY (CMID, Name, Country, City) — перечень автозаводов: ID, название, страна, город;
- PRODUCTION(PRID, CMID, CRID, Year, Month, Amount, Income) — производство автомобилей: ID, ID автозавода, ID автомобиля, год, месяц, количество выпущенных автомобилей, доход компании.

Содержимое таблиц этой базы данных следующее:

```
SELECT * FROM CAR;

 CRID TYPE
----- ------------
    1 автобус
    2 внедорожник
    3 г.автомобиль
    4 л.автомобиль
    5 микроавтобус
5 строк выбрано.
```

```
SELECT * FROM COMPANY;

 CMID NAME       COUNTRY    CITY
----- ---------- ---------- ----------------
    1 АЗЛК       Россия     Москва
    2 БЕЛАЗ      Беларусь   Жидино
    3 ВАЗ        Россия     Тольятти
    4 ГАЗ        Россия     Нижний Новгород
    5 ЗАЗ        Украина    Запорожье
    6 ЗИЛ        Россия     Москва
    7 КАМАЗ      Россия     Набережные Челны
    8 КРАЗ       Украина    Кременчуг
    9 ЛАЗ        Украина    Львов
   10 МАЗ        Беларусь   Минск
   11 ПАЗ        Россия     Павлово
   12 РАФ        Латвия     Елглава
   13 УАЗ        Россия     Ульяновск
   14 КАЗ        Грузия     Кутаиси
   15 ЛУАЗ       Украина    Луцк
15 строк выбрано.

SELECT * FROM PRODUCTION;

PRID CMID CRID  YEAR  MONTH  AMOUNT  INCOME
---- ---- ---- ----- ------ ------- -------
   1    1    4  2005      1      30      40
   2    2    3  2005      2      25      47
   3    3    4  2005      3      60      77
   4    3    2  2005      4      55      70
   5    4    3  2005      3      17      25
   6    4    4  2005      4      33      44
   7    4    1  2005      5      15      25
   8    5    4  2005      6      13      22
   9    6    3  2005      6      24      47
  10    6    1  2005      6      14      18
  11    7    3  2005      7      36      50
  12    7    3  2006     11      11      27
  13    8    3  2005      7      12      15
  14    9    1  2006      7      10      14
  15   10    3  2005      8      13      22
  16   10    1  2005      9      15      27
  17   11    1  2005     10       7      14
  18   12    1  2005     12       4      10
  19   13    4  2005     11      12      12
  20   13    2  2005     11      11      14
  21   13    1  2005     11       6      11
  22   13    3  2005     12       8      36
  23    6 NULL  NULL   NULL    NULL    NULL
  24    1 NULL  NULL   NULL    NULL    NULL
  25 NULL    1  NULL   NULL    NULL    NULL
  26 NULL    2  NULL   NULL    NULL    NULL
  27 NULL    3  NULL   NULL    NULL    NULL
  28   15 NULL  NULL   NULL    NULL
28 строк выбрано.
```

Так как большинство последующих запросов имеет отношение к автозаводам и выпускаемыми ими типами автомобилей, приведем графическую схему этой взаимосвязи, которая

облегчит проверку правильности приводимых далее запросов.

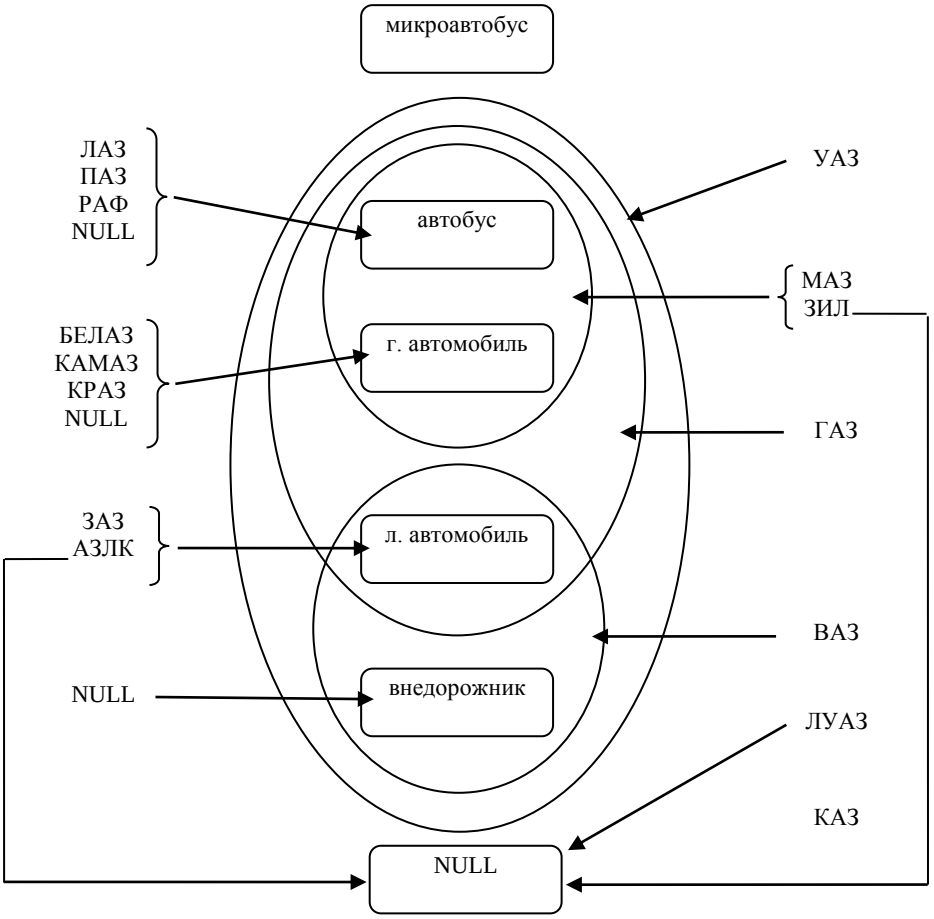


Рис. 2 Графическое представление взаимосвязей автозаводов и выпускаемых ими типов автомобилей

Прокомментируем этот рисунок.

- Отсутствие стрелок на тип автомобиля «микроавтобус» — этот тип автомобиля присутствует в базе данных, но ни один автозавод его не производит.
- Отсутствие стрелок из автозавода КАЗ — этот автозавод присутствует в базе данных, но отсутствуют сведения о производстве им автомобилей.
- Наличие `NULL` в списках автозаводов — в базе данных имеется информация о производстве того или иного типа автомобиля, но автозавод-производитель не известен.
- Наличие стрелок от автозаводов (АЗЛК, ЗИЛ, ЛУАЗ) на тип автомобиля `NULL` — в базе данных имеется информация о производстве такими автозаводами автомобилей, но каких именно — неизвестно.

2.2 Выражение предиката ⊇ посредством FORALL

Теперь давайте приведем пример выражения условия вхождения множества в другое множество с указанием трех вариантов: с использование предиката `CONTAINS` (⊇), с использованием квантора общности `FORALL` и с использованием только тех возможностей, которые имеются в стандартном `SQL` (предикат существования `EXISTS` и отрицания).

Запрос (теоретико-множественный). Вывести названия автозаводов, множество производимых типов автомобилей которых включает или равно множеству типов автомобилей, производимых автозаводом МАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  (SELECT Type
        FROM   CAR
        WHERE  CRID IN
          (SELECT CRID                  -- множество типов
           FROM   PRODUCTION p1         -- автомобилей, производимых
           WHERE  p1.CMID = c1.CMID))   -- искомой компанией
            CONTAINS                    -- СОДЕРЖИТ В СЕБЕ
       (SELECT Type                     -- множество типов
        FROM   CAR                      -- автомобилей, производимых
        WHERE  CRID IN                  -- автозаводом МАЗ
          (SELECT CRID
           FROM   PRODUCTION p2, COMPANY c2
           WHERE  p2.CMID = c2.CMID AND
                  c2.Name = 'МАЗ'));
```

В первом запросе фразы WHERE мы формируем множество типов автомобилей, производимых текущим автозаводом, зафиксированным во внешнем запросе. Во втором запросе формируется множество типов автомобилей, производимых заводом МАЗ.

Запрос (с квантором общности). Вывести названия таких автозаводов, которые производят как минимум ВСЕ те типы автомобилей, которые производятся автозаводом МАЗ.

Немного изменим формулировку запроса таким образом, чтобы она была очень близка к записи запроса с квантором общности:

Вывести названия таких автозаводов, что для ЛЮБОГО типа автомобиля, производимого заводом МАЗ, СУЩЕСТВАЕТ строка таблицы PRODUCTION, указывающая на то, что такой же тип автомобиля производится искомым заводом.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL                       -- ДЛЯ ВСЕХ
         (SELECT *                  -- производимых на заводе МАЗ
          FROM   PRODUCTION p2      -- типов автомобилей
          WHERE  p2.CMID IN
                 (SELECT CMID
                  FROM   COMPANY c2
                  WHERE  c2.Name = 'МАЗ')
                      ON
          EXISTS                    -- СУЩЕСТВУЕТ строка в PRODUCTION,
             (SELECT *              -- свидетельствующая,что тот же
              FROM  PRODUCTION p1 --тип авто производится на искомом
              WHERE p1.CMID = c1.CMID AND   -- заводе
                    p1.CRID = p2.CRID));
```

Давайте прокомментируем этот запрос. Сначала производится выбор очередной строки таблицы COMPANY, которой присваивается синоним (алиас) c1. В запросе предиката FORALL выбирается множество строк таблицы PRODUICTION, которые относятся к производству автомобилей заводом МАЗ. Теперь для каждой строки таблицы, сформированной этим запросом, то есть таблицы с алиасом p2, проверяется, существует ли такая строка в таблице PRODUICTION, которая имеет отношение к компании, зафиксированной в таблице с алиасом c1, и согласно которой осуществляется производство автомобиля того же типа, что и в строке, зафиксированной в таблице с алиасом p2. Если предикат EXISTS оказывается истинным для всех строк таблицы p2, то предикат FORALL принимает значение true и текущая строка таблицы COMPANY попадает в результат вычисления запроса. Если же предикат EXISTS оказывается ложным хотя бы для одной из строк таблицы p2, то предикат FORALL принимает значение false и текущая строка таблицы COMPANY не попадает в результат вычисления запроса.

2.3 Алгоритм преобразования FORALL в стандартный SQL

Теперь приведем алгоритм преобразования запроса с предикатом FORALL в запрос, допустимый в стандартном SQL. Он довольно простой и заключается в следующем:

- фраза FORALL заменяется на NOT EXISTS;
- условие, заданное во фразе ON, взятое с отрицанием (NOT), приписывается к фразе WHERE запроса фразы FORALL через логическую связку AND.

Схематически это выглядит следующим образом:

```
        FORALL (SELECT... FROM... WHERE условие_1 ON условие_2)
заменяется на
        NOT EXISTS (SELECT... FROM... WHERE условие_1 AND NOT условие_2)
```

Таким образом, можно переформулировать наш запрос следующим образом:

Запрос (с квантором существования и отрицаниями). Вывести названия автозаводов, для которых не существуют такие типы производимых автозаводом МАЗ типов автомобилей, которые бы они (искомые заводы) не производили.

Вот как выглядит этот запрос в результате применения к предыдущему запросу с квантором общности указанного выше алгоритма.

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
            (SELECT *
             FROM   PRODUCTION p2
             WHERE  p2.CMID IN (SELECT CMID
                                FROM   COMPANY c2
                                WHERE  c2.Name = 'МАЗ')
                       AND
                    NOT EXISTS (SELECT *
                                FROM   PRODUCTION p1
                                WHERE  p1.CMID = c1.CMID AND
                                       p1.CRID = p2.CRID));

NAME
-----
ГАЗ
ЗИЛ
МАЗ
УАЗ
4 строк выбрано.
```

Единственное, что надо было бы сделать в этом запросе, так это удалить из результата сам автозавод МАЗ.

Следует отметить, что если B является пустым множеством, то выражение $A \supseteq B$ является всегда истинным не зависимо от того, что собой представляет A. Это означает, что если запрос предиката FORALL возвращает пустое множество строк, то, то сам предикат FORALL оказывается истинным при любом условии фразы ON. Так, например, если в предыдущем запросе вместо МАЗ мы воспользуемся заводом, который ничего не выпускает или который вообще отсутствует в таблице COMPANY, то будет выведен весь список имеющихся в базе данных заводов:

Запрос. Вывести заводы, которые производят как минимум те типы автомобилей, что и завод КАЗ:

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID IN (SELECT CMID
                               FROM   COMPANY c2
                               WHERE  c2.Name = 'КАЗ')
                      AND
                   NOT EXISTS (SELECT *
                               FROM   PRODUCTION p1
                               WHERE  p1.CMID = c1.CMID AND
                                      p1.CRID = p2.CRID));

NAME
------
АЗЛК
БЕЛАЗ
ВАЗ
ГАЗ
ЗАЗ
ЗИЛ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
УАЗ
КАЗ
ЛУАЗ
15 строк выбрано.
```

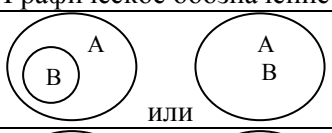
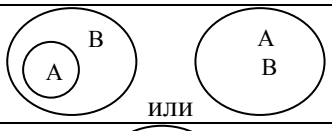
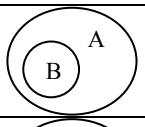
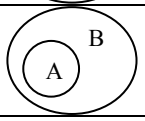
Такой же результат мы получили бы и в том случае, когда вместо завода КАЗ использовали бы несуществующий автозавод, например, XYZ.

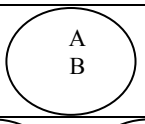
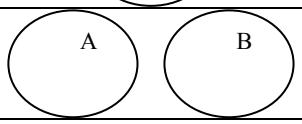
Об этом мы еще поговорим в следующем разделе при обсуждении предиката $\subseteq$.

3 Представление в SQL других предикатов над множествами

Рассмотрим теперь, как можно в SQL записывать выражения, предполагающие использование других теоретико-множественных предикатов. Для этого, прежде всего, приведем список этих предикатов с указанием их смысла.

Таблица 1. Перечень предикатов над множествами

Предикат	Смысл	Графическое обозначение
$A \supseteq B$	Множество A содержит множество B или равно ему	
$A \subseteq B$	Множество A содержится во множестве B или равно ему	
$A \supset B$	Множество A строго содержит множество B	
$A \subset B$	Множество A строго содержится во множестве B	

Предикат	Смысл	Графическое обозначение
A = B	Множества A и B равны	
A ∩ B = ∅	Множество A и B не пересекаются	

Для каждого из приведенных предикатов существует его отрицание, которые мы здесь не приводим.

Теперь последовательно рассмотрим, как выражаются эти предикаты в SQL. При этом, как и выше, мы будем приводить примеры во всех трех нотациях: теоретико-множественной, с квантором общности и с использованием стандартных возможностей.

3.1 Представление предиката ⊆

Итак, заменим в предыдущем примере предикат ⊇ на ⊆.

Запрос (теоретико-множественный). Вывести названия автозаводов, множество производимых типов автомобилей которых включается в или равно множеству типов автомобилей, производимых автозаводом МАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  (SELECT Type
        FROM   CAR
        WHERE  CRID IN (SELECT CRID
                        FROM   PRODUCTION p1
                        WHERE  p1.CMID = c1.CMID))
           IS CONTAINED IN
       (SELECT Type
        FROM   CAR
        WHERE  CRID IN (SELECT CRID
                        FROM   PRODUCTION p2, COMPANY c2
                        WHERE  p2.CMID = c2.CMID AND
                               c2.Name = 'МАЗ'));
```

Для теоретико-множественного описания никаких проблем не существует, заменяем обозначение одного предиката (CONTAINS) на другой (IS CONTAINED IN). Однако в записи с использованием квантора общности дело обстоит чуть сложнее.

Запрос (с квантором общности). Вывести названия таких автозаводов, которые производят не боле чем ВСЕ те типы автомобилей, которые производятся автозаводом МАЗ.

Опять изменим формулировку запроса, чтобы она была близка к записи запроса с квантором общности:

Вывести названия таких автозаводов, что для ЛЮБОГО типа производимого на этих автозаводах автомобиля, СУЩЕСТВАЕТ строка таблицы PRODUCTION, указывающая на то, что такой же тип автомобиля производится заводом МАЗ

Обратите внимание, что по сравнению с предыдущим запросом, ориентирующемся на предикат ⊇, здесь меняются местами выражения. Если раньше (в случае предиката ⊇) нам надо было, чтобы «для любого ТИПА АВТОМОБИЛЯ ПРОИЗВОДИМОГО ЗАВОДОМ МАЗ, существовал соответствующий тип автомобиля искомого завода», то теперь необходимо, чтобы «для любого ТИПА АВТОМОБИЛЯ ИСКОМОГО ЗАВОДА существовал соответствующий тип автомобиля завода МАЗ». То есть в этих высказываниях меняются местами посылки и заключения.

Итак, наш запрос выглядит следующим образом:

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID
                 ON
               EXISTS (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='МАЗ')));
```

Наконец, теперь не представляет никакого труда выразить этот запрос в терминах стандартного SQL.

Запрос (с квантором существования и отрицаниями). Вывести названия автозаводов, для которых не существуют такие типы производимых ими автомобилей, которые не производятся на заводе МАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM   COMPANY c2
                                         WHERE c2.Name='МАЗ')));
NAME
--------
БЕЛАЗ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
КАЗ
8 строк выбрано.
```

Если внимательно посмотреть на результат, то можно обнаружить, что в список попал завод КАЗ, который согласно нашей базе данных ничего не произвел. И в этом нет ничего удивительного, так как для этого завода запрос предиката FORALL возвращает пустое множество и поэтому этот предикат автоматически принимает значение true и, следовательно, КАЗ попадает в результат выполнения запроса. Также заметим, что хотя завод ЛУАЗ также по сути ничего не произвел согласно нашей базы данных, однако о нем все же имеется строка в таблице PRODUCTION, и поэтому этот завод в результат не попал.

Также можно обнаружить, что этот список не содержит еще один завод, который производит автобусы и грузовые автомобили, как и МАЗ, а именно, ЗИЛ. Все дело в том, что в нашей базе данных имеется информация, указывающая, что ЗИЛ производит еще что-то, но что именно, не известно (в таблице PRODUCTION имеется строка, относящаяся к заводу ЗИЛ, в которой в качестве значения CRID стоит NULL). Проблемы использования значений NULL в запросах рассматриваемого

типа будут обсуждены в одном из последующих разделов.

Давайте наложим дополнительное условие на искомые заводы.

Запрос. Вывести названия таких автозаводов, что для всех выпускаемых ими типов автомобилей справедливо условие: если год выпуска 2005, то такой тип также производится МАЗом.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID AND Year = 2005
                 ON
               EXISTS (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='МАЗ')));
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND Year = 2005 AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM   COMPANY c2
                                         WHERE c2.Name='МАЗ')));
NAME
----------
КРАЗ
РАФ
МАЗ
КАМАЗ
ЛАЗ
БЕЛАЗ
ПАЗ
ЗИЛ
КАЗ
ЛУАЗ
10 строк выбрано.
```

Условие запроса сформулировано таким образом, что проверка на год выпуска следует производить в посылке (выделено жирным шрифтом). Поэтому нет ничего удивительного, что в ответ попали, например, заводы ЛАЗ и ЛУАЗ, которые вообще ничего не произвели в 2005 году (запрос предиката `FORALL` возвращает пустое множество строк). Кроме того, для нас уже понятно, почему в списке также присутствует КАЗ.

Совсем другой результат мы получим в том случае, когда формулировка запроса требует размещения этого дополнительного условия в заключении предиката `FORALL`:

Запрос. Вывести названия таких автозаводов, которые произвели в 2005 году не боле `чем ВСЕ` те типы автомобилей, которые производятся автозаводом МАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID
                 ON
               (Year = 2005 AND
               EXISTS (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='МАЗ'))));
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
        (SELECT *
                FROM   PRODUCTION p2
                WHERE  p2.CMID = c1.CMID
                 AND NOT
               (Year = 2005 AND
               EXISTS (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='МАЗ'))));
NAME
-----
КРАЗ
РАФ
МАЗ
КАЗ
БЕЛАЗ
ПАЗ
6 строк выбрано.
```

Совет Всякий раз, когда имеются дополнительные условия, накладываемые на искомые строки таблиц, поймите их природу в том смысле, чтобы понять, где их следует размещать: в посылке или заключении предиката FORALL.

3.2 Представление предикатов = и ≠

Для представления запросов, требующих сравнения двух множеств на равенство (=) можно воспользоваться следующими эквивалентными выражениями над множествами:

$A = B \Leftrightarrow A \subseteq B \text{ AND } B \supseteq A$

Таким образом, приводимый далее запрос можно записать следующим образом:

Запрос. Вывести заводы, производящие такие и только такие типы автомобилей, что и КАМАЗ

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID IN (SELECT CMID
                               FROM   COMPANY c2
                               WHERE  c2.Name = 'КАМАЗ')
                      AND
                   NOT EXISTS (SELECT *
                               FROM   PRODUCTION p1
                               WHERE  p1.CMID = c1.CMID AND
                                      p1.CRID = p2.CRID))
       AND
       NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='КАМАЗ')));
NAME
------
КРАЗ
КАМАЗ
БЕЛАЗ
3 строк выбрано.
```

Предикат неравенства двух множеств получается как отрицание равенства. Теперь, если обратить внимание, что равенство выражается следующими двумя предикатами EXISTS (см. предыдущий пример)

```
NOT EXISTS (...) AND NOT EXISTS (...)
```

то неравенство принимает следующий вид:

```
NOT (NOT EXISTS (...) AND NOT EXISTS (...))⇔
EXISTS (...) OR EXISTS (...)
```

Таким образом, приводимый далее запрос можно записать следующим образом:

Запрос. Вывести заводы, множество производимых типов автомобилей которых отличается от множества типов автомобилей, производимых на КАМАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID IN (SELECT CMID
                               FROM   COMPANY c2
                               WHERE  c2.Name = 'КАМАЗ')
                      AND
                   NOT EXISTS (SELECT *
                               FROM   PRODUCTION p1
                               WHERE  p1.CMID = c1.CMID AND
                                      p1.CRID = p2.CRID))
       OR
       EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='КАМАЗ')));
NAME
-------
ЛУАЗ
КАЗ
АЗЛК
ВАЗ
ГАЗ
ЗАЗ
ЗИЛ
ЛАЗ
МАЗ
ПАЗ
РАФ
УАЗ
12 строк выбрано.
```

Обратите внимание, не смотря на то, что МАЗ, ЗИЛ, ГАЗ, УАЗ также производят грузовые автомобили, как и КАМАЗ, однако эти заводы дополнительно производят автомобили других типов, а КАМАЗ — нет. Поэтому они попали в этот список. В список попали также ЛУАЗ и КАЗ, которые ничего не производят. В свою очередь, БЕЛАЗ и КРАЗ производят только грузовые автомобили, поэтому они сюда не попали.

3.3 Представление предикатов ⊃ и ⊂

Предикаты $\supset$ и $\subset$ могут быть выражены через уже определенные следующим образом:

$A \supset B \Leftrightarrow A \supseteq$ AND $A \neq B$

$A \subset B \Leftrightarrow A \subseteq$ AND $A \neq B$

Запрос. Вывести названия таких автозаводов, которые производят ВСЕ те типы автомобилей, которые производятся автозаводом МАЗ, и обязательно еще какие-то.

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID IN (SELECT CMID
                               FROM   COMPANY c2
                               WHERE  c2.Name = 'МАЗ')
                      AND
                   NOT EXISTS (SELECT *
                               FROM   PRODUCTION p1
                               WHERE  p1.CMID = c1.CMID AND
                                      p1.CRID = p2.CRID))
         AND
        (EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID IN (SELECT CMID
                               FROM   COMPANY c2
                               WHERE  c2.Name = 'МАЗ')
                      AND
                   NOT EXISTS (SELECT *
                               FROM   PRODUCTION p1
                               WHERE  p1.CMID = c1.CMID AND
                                      p1.CRID = p2.CRID))
         OR
         EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM  COMPANY c2
                                         WHERE c2.Name='МАЗ'))));
NAME
-----
ГАЗ
ЗИЛ
УАЗ
3 строк выбрано.
```

Конечно, запрос выглядит довольно сложным, однако, зная, как представлять предикаты $\supseteq$ и $\subseteq$ через FORALL, а затем через EXISTS, написание такого сложного запроса, как приведенный выше, не требует чрезвычайных усилий. Это чисто механический процесс.

3.4 Представление предикатов пересекаемости и непересекаемости двух множеств

Относительно множеств определяется еще два предиката, которые устанавливают, пересекаются ли два множества или нет или, другими словами, имеются ли такие элементы, которые принадлежат обоим множествам. Формально это записывается следующим образом:

$A \cap B = \varnothing$ — множества не пересекаются

$A \cap B \neq \varnothing$ — множества пересекаются.

Здесь символ $\varnothing$ соответствует пустому множеству.

Рассмотрим сначала выражение, представляющее непересекаемость двух множеств, а затем с его помощью построим выражение пересекаемости множеств. Опять же, сначала выразим предикат непересекаемости через FORALL, а затем преобразуем полученное

выражение в стандартный SQL. Покажем это на примере.

Запрос. Вывести названия автозаводов, которые не производят автомобили таких типов, как МАЗ. Или, другими словами, вывести такие заводы, что любой производимый ими тип автомобиля не входит во множество автомобилей, производимых на МАЗе.

Вот как выглядит этот запрос с использованием FORALL:

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL                        -- любой производимый
         (SELECT *                   -- искомой компанией
          FROM   PRODUCTION p2       -- тип автомобиля
          WHERE  p2.CMID = c1.CMID
                 ON
          p2.CRID NOT IN             -- не входит во множество
             (SELECT CRID            -- типов автомобилей,
              FROM   PRODUCTION p1 -- производимых на МАЗе
              WHERE  p1.CMID IN (SELECT CMID
                                 FROM  COMPANY c2
                                 WHERE c2.Name='МАЗ')));
```

Теперь не представляет труда преобразовать этот запрос в стандартный SQL. Единственно, отметим, что отрицание условия после фразы ON производится заменой фразы NOT IN на IN:

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
               (SELECT *
                FROM   PRODUCTION p2
                WHERE  p2.CMID = c1.CMID
                  AND
                p2.CRID IN
                    (SELECT CRID
                     FROM   PRODUCTION p1
                     WHERE  p1.CMID IN (SELECT CMID
                                        FROM  COMPANY c2
                                        WHERE c2.Name='МАЗ')));
NAME
------
АЗЛК
ВАЗ
ЗАЗ
ЛУАЗ
КАЗ
5 строк выбрано.
```

Предикат пересекаемости двух множеств является отрицанием предиката непересекаемости. Поэтому простым преобразованием предыдущего SQL–запроса мы получаем выражение для следующего запроса:

Запрос. Вывести названия автозаводов, которые производят хотя бы один такой тип автомобиля, как МАЗ.

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS
         (SELECT *
          FROM   PRODUCTION p2
          WHERE  p2.CMID = c1.CMID
            AND
          p2.CRID IN
              (SELECT CRID
               FROM   PRODUCTION p1
               WHERE  p1.CMID IN (SELECT CMID
                                  FROM  COMPANY c2
                                  WHERE c2.Name='МАЗ')));
NAME
------
БЕЛАЗ
ГАЗ
ЗИЛ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
УАЗ
10 строк выбрано.
```

Единственное отличие этого SQL–запроса от предыдущего заключается в том, что мы во фразе WHERE внешнего запроса заменили предикат NOT EXISTS на EXISTS.

Обратите внимание, что вроде бы объединение результатов двух последних запросов должно дать множество всех автозаводов. Но это не так в связи с тем, что автозаводы, не производящие никаких автомобилей, не попадают ни в одно из этих множеств (в данном случае это автозавод КАЗ).

4 Дополнительные аспекты

В этом разделе мы рассмотрим варианты, когда запрос предиката FORALL не содержит фразы WHERE, содержит фразы GROUP BY и HAVING, использование фразы ALL стандарта SQL для выражения некоторых видов предикатов над множествами, а также особенности использования значений NULL в предикатах над множествами.

4.1 Отсутствие фразы WHERE в запросе предиката FORALL

Возможны ситуации, когда запрос предиката FORALL не содержит фразы WHERE. В этом случае при преобразовании FORALL-запроса в запрос стандарта SQL эта фраза все же вставляется и к ней приписывается условие фразы ON с отрицанием, но уже без логической связки AND, как это показано в следующей формальной записи:

FORALL (SELECT… FROM… ON условие)

преобразуется в:

NOT EXISTS (SELECT... FROM… WHERE NOT условие)

Приведем пример.

Запрос. Вывести названия автозаводов, которые выпускали автомобили во все годы.

Примечание. Пусть для нас критерием «всех» годов является перечень годов, которые имеются в таблице PRODUCTION.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT *
                FROM   PRODUCTION p2
                  ON
                       EXISTS (SELECT *
                               FROM   PRODUCTION p3
                               WHERER p3.CMID = c1.CMID AND
                                      p3.Year = p2.Year);
```

Это запрос на предикат ⊇, который мы рассмотрели самым первым. Однако здесь нас интересуют все года, поэтому в этом случае в запросе предиката FORALL не нужна фраза WHERE, которая отбирала бы строки, имеющие отношение к диапазону годов.

Приведя запрос к стандарту SQL, получим:

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT *
                   FROM   PRODUCTION p2
                   WHERE  NOT EXISTS (SELECT *
                                      FROM    PRODUCTION p3
                                      WHERE   p3.CMID=c1.CMID AND
                                              p3.Year=p2.Year));
строки не выбраны.
```

Хотя такой завод у нас есть — КАМАЗ, однако он не указан в ответе, так как опять же все дело в наличии в таблице PRODUCTION строк, указывающих на производство автомобилей с Year равным NULL. В разделе, посвященном учету значений NULL, мы добьемся того, чтобы в этом запросе КАМАЗ выводился.

4.2 Использование фраз GROUP BY и HAVING в запросе предиката FORALL

В общем случае запрос предиката FORALL может содержать фразы GROUP BY и HAVING, то есть предикат имеет вид:

```
FORALL (SELECT... FROM... [WHERE...] [GROUP BY...] [HAVING...]
        ON условие)
```

Синтаксис и семантика фраз GROUP BY и HAVING стандартные.

При наличии этих фраз семантика предиката FORALL следующая.

- Если имеется фраза WHERE, то прежде всего отбираются строки, удовлетворяющие ее условию.
- Если присутствует фраза GROUP BY без HAVING, то формируются группы строк запроса. Затем, если на всех группах оказывается истинным условие фразы ON, то предикат FORALL истинен; в противном случае он ложен.
- Если присутствует фраза HAVING без GROUP BY, то формируется одна группа, состоящая из всех строк запроса (с учетом возможной фразы WHERE). Если на этой группе условие фразы HAVING оказывается ложным, предикат FORALL становится истинным. Если же условие фразы HAVING истинно, то проверяется истинность условия фразы ON, если оно истинно, то предикат FORALL истинен, в противном случае он ложен.
- Если присутствуют фразы GROUP BY и HAVING, то формируются группы строк запроса согласно фразе GROUP BY. Затем на каждой группе строк проверяются истинность условий фраз HAVING и ON. Если на всех группах эти условия истинны, то предикат FORALL истинен, в противном случае он ложен. Проверка истинности условий фраз HAVING и ON на группе производится так же, как и в предыдущем случае, а именно:

а) если на группе условие HAVING ложно, то на самой группе условие истинно;

б) если на группе условие HAVING истинно, то при истинности условия фразы ON, то в целом на группе условие истинно; в противном случае оно ложно.

Следует отметить, что при наличии фраз GROUP BY и/или HAVING условие фразы ON должно быть таким, которое допустимо во фра-

зе HAVING. Это объясняется тем, что это условие проверяется на группах, а не на отдельных строках. При отсутствии фраз GROUP BY и HAVING условие фразы ON может быть таким, которое допускается во фразе WHERE или HAVING. В первом случае мы имеем тот вариант предиката FORALL, который мы обсуждали в предыдущих разделах. Во втором случае по умолчанию предполагается наличие фразы HAVING (без GROUP BY).

В качестве примера рассмотрим следующий запрос.

Запрос. Вывести названия таких автозаводов, что не смотря на тот факт, что суммарные выпуски автомобилей в каждом из месяцев 2005 года были меньше 30, однако суммарный доход в каждом из месяцев 2005 превышал 60.

(Под всеми месяцами подразумевается не все 12 месяцев в году, а все те месяцы, в которые компания производила продукцию согласно таблице PRODUCTION).

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID AND
                      Year = 2005
               GROUP BY MONTH
               HAVING SUM(Amount) < 30
                 ON
               SUM(Income) > 35);
```

Обратим ваше внимание на факт использования константы 1 во фразе SELECT. В предыдущих разделах на этом месте мы использовали символ «*». Так можно было поступить, так как в запросе выбирались строки. В нашем же случае использование «*» было бы синтаксически неправильным. Нам надо зафиксировать факт того, что та или иная группа удовлетворяет соответствующему условию. Для этого следует во фразе SELECT использовать константу любого типа. Затем по наличию констант можно определить все ли группы удовлетворяют условию (тем более, что при преобразовании этого выражения в стандарт SQL нам надо будет делать проверку на существование таких групп).

Теперь покажем, как запросы такого вида перевести в стандартный SQL

```
FORALL (SELECT... FROM... WHERE... GROUP BY...
 HAVING условие_1 ON условие_2)
```

преобразуется в

```
NOT EXISTS (SELECT... FROM... WHERE... GROUP BY...
 HAVING условие_1 AND NOT условие_2)
```

Таким образом, наш предыдущий запрос приобретает следующий вид:

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID AND
                          Year = 2005
                   GROUP BY MONTH
                   HAVING SUM(Amount) < 30 AND
                          NOT SUM(Income) > 60);
```

```
NAME
----------
ЛУАЗ
КАЗ
АЗЛК
```

```
ВАЗ
ЗИЛ
КАМАЗ
ЛАЗ
7 строк выбрано.
```

Теперь приведем формальные правила преобразования других вариантов фразы FORALL.

FORALL (SELECT... FROM... WHERE... GROUP BY... ON условие)
преобразуется в
NOT EXISTS (SELECT... FROM... WHERE... GROUP BY... HAVING NOT условие).

FORALL (SELECT... FROM... WHERE... HAVING условие_1 ON условие_2)
преобразуется в
NOT EXISTS (SELECT... FROM... WHERE... HAVING условие_1 AND NOT условие_2).

FORALL (SELECT... FROM... WHERE условие_1 ON условие_2)
преобразуется в
NOT EXISTS (SELECT... FROM... WHERE условие_1 HAVING NOT условие_2)
если условие_2 соответствует правилам формирования условий фразы HAVING (например, имеются агрегатные функции), или преобразуется в
NOT EXISTS (SELECT... FROM... WHERE условие_1 AND NOT условие_2)
если условие_2 соответствует правилам формирования условий фразы WHERE.

Приведем несколько примеров.

Запрос. Вывести названия таких автозаводов, что суммарный доход во всех месяцах 2005 превышает 35.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID AND
                      Year = 2005
               GROUP BY MONTH
                 ON
               SUM(Income) > 35);

SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID AND
                          Year = 2005
                   GROUP BY MONTH
                   HAVING NOT SUM(Income) > 35);

NAME
--------
ЛУАЗ
КАЗ
АЗЛК
БЕЛАЗ
ВАЗ
ЗИЛ
```

```
КАМАЗ
ЛАЗ
УАЗ
9 строк выбрано.
```

Показательным в этом ответе является то, что тут присутствует УАЗ. На этом заводе было три выпуска автомобилей в 11 месяце в 2005 году, доход каждого из которых не превышал 14, однако суммарный доход составил 37 (кроме того, был выпуск в 12 месяце и он составил 36). ЛАЗ попал в результирующий список в связи с тем, что он не выпустил автомобили в 2005 г. (запрос фразы FORALL возвращает пустое множество строк). То же самое имеет место и по отношению к ЛУАЗ и КАЗ.

Запрос. Вывести названия автозаводов, суммарный выпуск автомобилей которых меньше 10 (вернее так: если они что-то производили, то суммарный выпуск был меньше 10).

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID
                 ON
               SUM(Amount) < 10);

SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID
                   HAVING NOT SUM(Amount) < 10);

NAME
------
ЛУАЗ
КАЗ
ПАЗ
РАФ
4 строк выбрано.
```

Если нас не интересуют автозаводы, которые ничего не произвели, то этот запрос может быть представлен без FORALL (NOT EXISTS) следующим образом:

```
SELECT   Name
FROM     COMPANY c1, PRODUCTION p1
WHERE    p1.CMID = c1.CMID AND Year = 2005
GROUP BY Name
HAVING   SUM(Amount) < 10;

NAME
------
ПАЗ
РАФ
2 строк выбрано.
```

Запрос. Вывести названия автозаводов, суммарный выпуск автомобилей которых в 2005 году меньше 10 (вернее так: если они что-то и произвели в 2005 г, то суммарный выпуск был меньше 10):

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID AND Year = 2005
                 ON
               SUM(Amount) < 10);

SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID AND Year = 2005
                   HAVING NOT SUM(Amount) < 10);
NAME
------
ЛУАЗ
КАЗ
ЛАЗ
ПАЗ
РАФ
5 строк выбрано.
```

ЛАЗ попал в результат, потому что он не произвел в 2005 году автомобилей, а ЛУАЗ и КАЗ – потому что они ничего не произвели.

Если нас не интересуют автозаводы, которые ничего не произвели в 2005 году, то этот запрос может быть представлен без FORALL (NOT EXISTS) следующим образом:

```
SELECT   Name
FROM     COMPANY c1, PRODUCTION p1
WHERE    p1.CMID = c1.CMID AND Year = 2005
GROUP BY Name
HAVING   SUM(Amount) < 10;

NAME
------
ПАЗ
РАФ
2 строк выбрано.
```

Как видим, здесь ЛАЗ, ЛУАЗ и КАЗ отсутствуют.

Запрос. Вывести названия таких автозаводов, что если в каждом месяце 2005 года суммарный выпуск автомобилей превышает 10, то суммарный доход ниже 20.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID AND
                      Year = 2005
               GROUP BY MONTH
               HAVING SUM(Amount) > 10
                 ON
               SUM(Income) < 20);
```

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID
                          AND Year = 2005
                   GROUP BY MONTH
                   HAVING SUM(Amount) > 10 AND
                          NOT SUM(Income) < 20);

NAME
-------
ЛУАЗ
КАЗ
КРАЗ
ЛАЗ
ПАЗ
РАФ
6 строк выбрано.
```

Наконец, приведем еще один запрос

Запрос. Вывести названия таких автозаводов, что если у них был выпуск автомобилей в 2005 году, то суммарный доход в этом году превышает 35.

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT 1
               FROM   PRODUCTION p1
               WHERE  p1.CMID = c1.CMID AND
                      Year = 2005
                 ON
               SUM(Income) > 100);

SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT 1
                   FROM   PRODUCTION p1
                   WHERE  p1.CMID = c1.CMID AND
                          Year = 2005
                   HAVING NOT SUM(Income) > 100);

NAME
-------
ЛУАЗ
КАЗ
ВАЗ
ЛАЗ
4 строк выбрано.
```

4.3 Выражение предикатов над множествами с помощью стандартного предиката ALL

В языке SQL имеется предикат ALL, который является ограниченной формой квантора общности. Напомним, что он имеет следующий обобщенный формат:

выражение_1 оператор_сравнения ALL (запрос)

Если раскрыть возможное содержимое запроса, то получим следующий формат:

выражение_1 оператор_сравнения ALL
(SELECT выражение_2 FROM таблица
[WHERE условие_1]
[GROUP BY условие_2]
[HAVING условие_3])

Этот предикат следующим образом выражается с помощью предиката FORALL:

FORALL (SELECT {* | константа}
[WHERE условие_1]
[GROUP BY условие_2]
[HAVING условие_3]
ON
выраж_1 оператор_сравн. выраж_2)

где константа используются при наличии фраз GROUP BY (см. описание использование этой фразы в предикате FORALL).

Приведем пример выражения запроса с помощью этих двух предикатов.

Запрос. Вывести название автозавода, который получил самую большую прибыль за производства одного типа автомобиля в июле 2005 года.

Вот как выглядит этот запрос с использованием FORALL:

```
SELECT c1.Name
FROM   COMPANY c1, PRODUCTION p1
WHERE  c1.CMID = p1.CMID AND Year = 2005 AND Month = 7 AND
       FORALL (SELECT *
               FROM PRODUCTION p2
               WHERE Year = 2005 AND Month = 7 AND
                           p1.CMID != p2.CMID
                    ON
               p1.Income > p2.Income);
```

Эквивалентное ему выражение с использованием ALL:

```
SELECT c1.Name
FROM   COMPANY c1, PRODUCTION p1
WHERE  c1.CMID = p1.CMID AND Year = 2005 AND Month = 7 AND
       p1.Income > ALL (SELECT p2.Income
                        FROM PRODUCTION p2
                        WHERE Year = 2005 AND Month = 7 AND
                              p1.CMID != p2.CMID);

NAME
--------
КАМАЗ
```

4.4 Учет значений NULL

Мы уже могли убедиться, что наличие значений NULL может приводить не к тем результатам, которые ожидалось получить. Так, например, в запросе на вывод названий автозаводов, выпускавших автомобили во все годы, мы не получили КАМАЗ, который вроде бы выпускал автомобили во все годы. Все дело в том, что для нас критерием ВСЕХ годов является перечень годов, присутствующих в столбце Year таблицы PRODUCTION. Этот столбец содержит значения NULL, которое рассматривается в SQL в качестве самостоятельного значения, хотя оно, по сути, означает «значение отсутствует». Таким образом, если значение NULL следует исключить из области рассмотрения, то это следует сделать в запросе явно с помощью предиката IS NOT NULL, как это показано в следующем варианте этого запроса (дополнительный предикат приведен жирным шрифтом).

```
SELECT Name
FROM   COMPANY c1
WHERE  FORALL (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.Year IS NOT NULL AND
                 ON
                    EXISTS (SELECT *
                            FROM   PRODUCTION p3
                            WHERER p3.CMID = c1.CMID AND
                                   p3.Year = p2.Year);

SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS (SELECT *
                   FROM   PRODUCTION p2
                   WHERE  p2.Year IS NOT NULL AND
                          NOT EXISTS (SELECT *
                                      FROM   PRODUCTION p3
                                      WHERE  p3.CMID=c1.CMID AND
                                             p3.Year=p2.Year));

NAME
------
КАМАЗ
```

Обратите внимание, отсечение ненужных строк производится в посылке предиката FORALL, так как именно здесь надо указать, что следует принимать во внимание только те строки, в которых год не равен NULL.

То же самое относится к ранее рассмотренному запросу: «Вывести названия таких автозаводов, которые производят не более чем ВСЕ те типы автомобилей, которые производятся автозаводом МАЗ» Для того, чтобы в результат попал ЗИЛ, SQL-запрос следует записать следующим образом.

```
SELECT Name
FROM   COMPANY c1
WHERE  NOT EXISTS
            (SELECT *
             FROM   PRODUCTION p2
             WHERE  p2.CMID = c1.CMID AND CRID IS NOT NULL AND
                    NOT EXISTS
                       (SELECT *
                        FROM  PRODUCTION p1
                        WHERE p1.CRID = p2.CRID AND
                              p1.CMID IN (SELECT CMID
                                          FROM   COMPANY c2
                                          WHERE c2.Name='МАЗ')));

NAME
--------
БЕЛАЗ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
ЗИЛ
```

```
ЛУАЗ
КАЗ
10 строк выбрано.
```

Обратите внимание, что в ответе появился ЗИЛ. Однако остался КАЗ, который ничего не производит, и появился ЛУАЗ, для которого теперь посылка предиката FORALL дает пустое множество.

4.5 Учет пустого множества в запросе предиката FORALL

Во многих приведенных выше запросах мы не хотели бы выводить те значения автозаводов, для которых запрос предиката FORALL формирует пустое множество, так как в этих случаях выводимые значения не всегда соответствуют тому, что хотелось бы получить. Например, в последнем запросе предыдущего раздела в ответ попали КАЗ и ЛУАЗ так как для них посылка предиката FORALL равна пустому множеству. Как бороться с этим явлением. Далее дается вариант решения этой задачи. Мы не уверенны, что это наилучший способ ее решения, тем не менее он ее решает.

Суть решения заключается в том, чтобы во фразе WHERE внешнего запроса дополнительно с предикатом FORALL использовать предикат EXISTS, который проверяет, чтобы запрос предиката FORALL был не пустым. То есть внешний запрос имеет следующий вид:

```
SELECT ... FROM ...
WHERE EXISTS (запрос) AND
      FORALL (запрос ON условие)
```

причем запрос в EXISTS и FORALL один и тот же.

Приведем примеры ранее использовавшихся запросов, в которых учтено это дополнение.

Запрос. Вывести названия таких автозаводов, которые производят не более чем ВСЕ те типы автомобилей, которые производятся автозаводом МАЗ. Если заводы не производят никаких автомобилей, то их не включать в результат:

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID) AND
       NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID AND
                   NOT EXISTS
                      (SELECT *
                       FROM  PRODUCTION p1
                       WHERE p1.CRID = p2.CRID AND
                             p1.CMID IN (SELECT CMID
                                         FROM   COMPANY c2
                                         WHERE c2.Name='МАЗ')));
NAME
----------
БЕЛАЗ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
7 строк выбрано.
```

Итак, мы избавились от КАЗ и ЛУАЗ. А теперь добавим заводы, которые удовлетворяют условию запроса и, дополнительно, производят еще какие-то автомобили, но какие, не известно (то есть у них в тип производимого автомобиля равен NULL).

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS
              (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID AND CRID IS NOT NULL) AND
             NOT EXISTS
              (SELECT *
               FROM   PRODUCTION p2
               WHERE  p2.CMID = c1.CMID AND CRID IS NOT NULL AND
                      NOT EXISTS
                         (SELECT *
                          FROM  PRODUCTION p1
                          WHERE p1.CRID = p2.CRID AND
                                p1.CMID IN (SELECT CMID
                                            FROM   COMPANY c2
                                            WHERE c2.Name='МАЗ')));
NAME
----------
БЕЛАЗ
КАМАЗ
КРАЗ
ЛАЗ
МАЗ
ПАЗ
РАФ
ЗИЛ
8 строк выбрано.
```

Таким образом, мы добавили в ответ ЗИЛ.

Запрос. Вывести названия таких автозаводов, что для всех выпускаемых ими типов автомобилей справедливо условие: если год выпуска 2005, то такой тип также производится МАЗом. Если завод ничего не произвел в 2005 году, то его не выводить.

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS
          (SELECT *
           FROM   PRODUCTION p2
           WHERE  p2.CMID = c1.CMID AND Year = 2005) AND
       NOT EXISTS
          (SELECT *
           FROM   PRODUCTION p2
           WHERE  p2.CMID = c1.CMID AND Year = 2005 AND
                  NOT EXISTS
                     (SELECT *
                      FROM  PRODUCTION p1
                      WHERE p1.CRID = p2.CRID AND
                            p1.CMID IN (SELECT CMID
                                        FROM   COMPANY c2
                                        WHERE c2.Name='МАЗ')));
```

```
NAME
----------
КРАЗ
РАФ
МАЗ
КАМАЗ
БЕЛАЗ
ПАЗ
ЗИЛ
7 строк выбрано.
```

Запрос. Вывести названия автозаводов, которые не производят автомобили таких типов, как МАЗ. Или, другими словами, вывести такие заводы, что любой производимый ими тип автомобиля не входит во множество автомобилей, производимых на МАЗе. Если завод ничего не производит, то его не выводить.

```
SELECT Name
FROM   COMPANY c1
WHERE  EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID) AND
       NOT EXISTS
           (SELECT *
            FROM   PRODUCTION p2
            WHERE  p2.CMID = c1.CMID
              AND
            p2.CRID IN
                (SELECT CRID
                 FROM   PRODUCTION p1
                 WHERE  p1.CMID IN (SELECT CMID
                                    FROM  COMPANY c2
                                    WHERE c2.Name='МАЗ')));
NAME
------
АЗЛК
ВАЗ
ЗАЗ
ЛУАЗ
4 строк выбрано.
```

По сравнению с предыдущим вариантом мы избавились от КАЗ.

Итак, как следует из изложенного материала, стандартный SQL обладает полнотой по выражению теоретико-множественных предикатов.

Литература

1. Claudio Fratarcangefi. Technique for Universal Quantification in SQL. SIGMOD RECORD, Vol. 20, No. 3, September 1991. – pp. 16 – 24.

Сведения об авторе:

Резниченко Валерий Анатольевич - к.ф-.м.н., с.н.с., ведущий научный сотрудник Института программных систем НАН Украины, область научных интересов - информационные системы, базы данных и знаний, электронные библиотеки
E-mail: vreznichenko_47@mail.ru

Статья поступила в редакцию 15.08.2011 г.