УДК 519.17

ВЗГЛЯД НА МАТЕМАТИЧЕСКИЕ ВЫЧИСЛЕНИЯ СРЕДСТВАМИ БАЗ ДАННЫХ НА ПРИМЕРЕ УМНОЖЕНИЯ И ТРАНСПОНИРОВАНИЯ МАТРИЦ

© В.Ю. Маркеев, А.А. Крючков

Ключевые слова: математические вычисления; умножение матриц; транспонирование матриц; базы данных; хранимые процедуры.

Рассмотрена вертикальная структура представления матриц внутри базы данных. Показана реализация операций транспонирования и умножения матриц средствами баз данных.

В [1] было показано, что вопреки распространенному мнению о применимости баз данных не более чем для длительного хранения информации средствами баз данных можно вполне успешно производить математические вычисления. Данное утверждение было проиллюстрировано на примере алгоритма Дейкстры, применяемого для нахождения кратчайшего пути между вершинами взвешенного графа. Экспериментально было установлено, что реализация алгоритма Дейкстры средствами СУБД MySQL для графов большой размерности (десятки тысяч вершин и выше) по скорости выполнения превосходит классическую матричную реализацию данного алгоритма средствами языка C++.

Что представляется немаловажным для авторов данной статьи, возможности реляционной модели организации данных при проектировании схемы БД позволяют выразить особенности сущностей и отношений между ними на языке предметной области, что при последующем использовании языка SQL, посредством которого производятся запросы к базе данных, придает алгоритмической реализации наглядность и изящество. А именно, программный код SQL-запросов и хранимых процедур при применении «говорящих» названий таблиц, полей внутри таблиц и переменных становится настолько самодокументируемым, что текст запроса практически дословно отражает его описание на языке предметной области.

Рассмотрим реализацию операций транспонирования и умножения действительных матриц, в т. ч. разреженных, средствами базы данных MySQL. Схему данных для матричных вычислений средствами БД представляется удобным организовать следующим образом (рис. 1).

Каждая запись таблицы ***matrix*** соответствует конкретной матрице, для которой в данной реализации достаточно знать непосредственно только ее размерность, которая выражена в полях *height* и *width*.

Определенный элемент $\mathbf{A}_{ij}$ некоторой матрицы **A** соответствует записи таблицы ***matrix_element***, первичный ключ которой состоит из трех характеристик: номера (*id*) матрицы *matrix*, к которой относится данный элемент, номера строки *i* и номера столбца *j*, отражающих положение элемента внутри данной матрицы. Значением элемента матрицы выступает поле *value* соответствующей записи таблицы ***matrix_element***.

Данный подход в теории проектирования баз данных принято называть *вертикальной организацией БД*. Горизонтальная же организация данных в данном случае матриц выглядела бы как двумерный массив. Если в языках и средах, подобных C++, представление матрицы в виде двумерного массива есть общепринятая практика, то попытки применения подобного подхода в БД, мягко говоря, неудобны и нецелесообразны.

В нашем случае вертикальная структура представления данных о матрицах легко допускает возможность хранения разреженных матриц по определению: т. е. если некоторые элементы матрицы равны нулю, то можно их не хранить в таблице ***matrix_element***; и обратно, если для некоторых элементов матрицы нет записей в таблице ***matrix_element***, то их следует считать равными нулю.

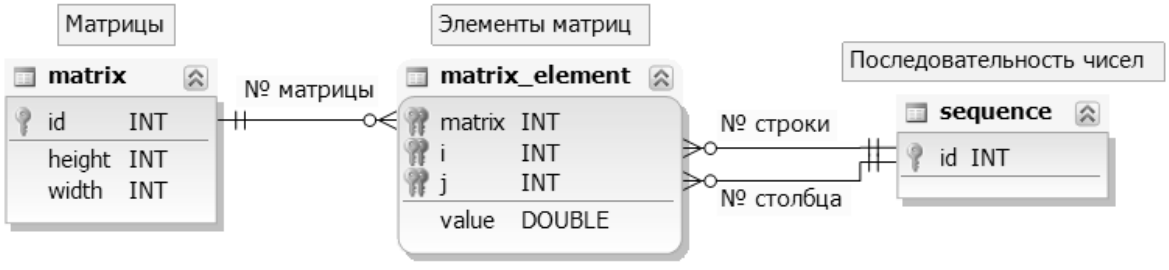

Рис. 1. Схема представления разреженных матриц в БД

Стоит отметить таблицу ***sequence***, состоящую из единственного поля *id*, записи которой представляют ряд натуральных чисел. Количество записей в таблице ***sequence*** должно быть не меньше, чем наибольшее из измерений (height и width), участвующих в вычислениях матриц. Назначением таблицы ***sequence*** является имитация целочисленных осей координат, которые потребуются при вычислениях. Отметим, что для увеличения производительности на поля таблиц, по которым будет осуществляться связывание (внешние ключи), были установлены индексы.

Для вывода вертикального представления матрицы будет использован следующий SQL-запрос:

```
select
  h.id i
 , w.id j
 , ifnull(a.value, 0) value
from matrix
join sequence h on h.id <= matrix.height
join sequence w on w.id <= matrix.width
 left join matrix_element a
  on a.matrix = matrix.id
  and a.i = h.id
  and a.j = w.id
where matrix.id = _a
order by h.id, w.id;
```

Изначально подключаем интересующую нас матрицу $\mathbf{c}_{id}$, равную параметру **_a**, затем с учетом ее размерности строится двумерное пространство независимыми осями *h* и *w*. На данном этапе при независимом соединении таблиц – «координатных осей» мы получили декартово произведение векторов *h* и *w*, что даст нам ровно *matrix.height * matrix.width* элементов в выборке независимо от того, сколько элементов матрицы присутствует в таблице ***matrix_element***. Далее, для каждого элемента пространства присоединяем соответствующую запись таблицы ***matrix_element*** посредством *LEFT JOIN*, что в случае отсутствия искомой записи сохранит целостность результирующей выборки, причем в качестве значения элемента, как, впрочем, и всех полей строки ***matrix_element***, будет выступать значение *null*. Для удобства ряд значений упорядочивается по строкам, а внутри них – по столбцам. В результирующую выборку включаются номера строк и столбцов матрицы, а также значения находящихся на их пересечении элементов. Для обработки отсутствующих (нулевых) элементов применяется встроенная функция *IFNULL*(), которая вместо отсутствующих *null*-значений подставит нуль.

Для обеспечения привычного табличного представления матрицы, SQL-запрос несколько изменится:

```
select
 group_concat(ifnull(a.value, 0) order by w.id separator '\t')
matrix
from matrix
join sequence h on h.id <= matrix.height
join sequence w on w.id <= matrix.width
 left join matrix_element a
  on a.matrix = matrix.id
  and a.i = h.id
  and a.j = w.id
where matrix.id = _a
group by h.id;
```

В данном случае выборка элементов разобьется на классы эквивалентности по признаку принадлежности элементов к одноименным строкам, после чего для каждого класса эквивалентности будет отображено по одной записи, внутри которой элементы данного класса (принадлежащие данной строке) будут выведены в горизонтальный ряд, разделяясь символом табуляции. Таким образом, вертикальное представление матрицы будет сведено к классической форме.

Задача транспонирования заданной матрицы стандартным образом решается путем двух вложенных циклов, которые, пробегая одну из половин двумерного массива, полученных разбиением матрицы по главной диагонали, меняют местами симметричные элементы. Также возникает проблема переопределения размерности массива при неравных количествах строк и столбцов матрицы. Что касается средств баз данных, то задача транспонирования решается аналогично выводу, за тем лишь исключением, что индексы строк и столбцов в результирующей выборке меняются местами. Транспонирование основано на следующем SQL-запросе:

```
select
  w.id i
 , h.id j
 , ifnull(a.value, 0) value
from matrix
join sequence h on h.id <= matrix.height
join sequence w on w.id <= matrix.width
 left join matrix_element a
  on a.matrix = matrix.id
  and a.i = h.id
  and a.j = w.id
where matrix.id = _a
order by w.id, h.id;
```

Особенно изящно средствами баз данных выполняется операция умножения матриц посредством единственного SQL-запроса, в то время как классический подход подразумевает три вложенных цикла. Напомним, что элементом результирующей матрицы $\mathbf{c}_{ij}$ будет сумма попарных произведений элементов *i*-й строки матрицы **A** и *j*-го столбца матрицы **B**, т. е. $\mathbf{c}_{ki} = \sum_k \mathbf{a}_{ik}\mathbf{b}_{kj}$. Далее приведен полный текст хранимой процедуры, производящей умножение матриц **_a** и **_b**, номера которых переданы в качестве входных параметров:

```
CREATE PROCEDURE math.mult(in _a INT, in _b int)
 SQL SECURITY INVOKER
BEGIN
 declare ah, aw, bh, bw, m INT;
 select
  matrix.height
 , matrix.width
 into ah, aw
 from matrix
 where matrix.id = _a;

 select
  matrix.height
 , matrix.width
 into bh, bw
 from matrix
 where matrix.id = _b;

 if aw = bh then
  # Позаботиться, чтобы в последовательности хватило
чисел
```

```
 call sequence(GREATEST(ah, aw, bw));

 insert into matrix set
  height = ah, width = bw;
 set m = LAST_INSERT_ID();

 insert into matrix_element
 select
   null
  , m
  , h.id
  , w.id
  , sum(ifnull(a.value, 0)*ifnull(b.value, 0)) value
 from sequence h
 join sequence w on w.id <= bw
  join sequence k on k.id <= aw
  left join matrix_element a
   on a.matrix = _a
   and a.i = h.id
   and a.j = k.id
  left join matrix_element b
   on b.matrix = _b
   and b.i = k.id
   and b.j = w.id
 where h.id <= ah
 group by h.id, w.id
 having value != 0;
else
 select 'incorrect length' error;
end if;
END
```

Особенно примечателен запрос на выборку элементов для их добавления в новую матрицу – результат умножения. Здесь к двумерному пространству новой матрицы добавляется третье (промежуточное) измерение, элементами которого будут произведения соответствующих элементов, которые затем при группировке сожмутся обратно в двумерное пространство путем суммирования элементов-значений внутри классов эквивалентности. Классами эквивалентности в данном случае выступают элементы новой (результирующей) матрицы. Блок «*having value*! = 0» исключит из результирующей выборки нулевые значения, которые нет необходимости сохранять в связи с вышеописанными договоренностями хранения разреженных матриц.

Математические вычисления средствами баз данных представляют значительный интерес авторов по ряду причин. Во-первых, это впечатляющие результаты устойчивости и производительности при решении задач огромных размерностей. Во-вторых, из эстетических соображений, т. к., по нашему мнению, это красиво и удобно. И наконец, данная область мало исследована, по крайней мере, ранее нам не доводилось встречать научных (и вообще каких-либо) работ на стыке математических вычислений и баз данных, что придает еще больший интерес дальнейшим исследованиям в данной области.

ЛИТЕРАТУРА

1. *Маркеев В.Ю., Крючков А.А.* Использование средств баз данных для математических вычислений на примере алгоритма Дейкстры // Вестник Тамбовского университета. Серия Естественные и технические науки. Тамбов, 2012. Т. 17. Вып. 4. С. 1234-1238.
2. *Беллман Р.* Введение в теорию матриц. М.: Наука. 1969.
3. *Гантмахер Ф.Р.* Теория матриц. М.: Физматлит, 2010. 560 с.

Поступила в редакцию 23 ноября 2012 г.

Markeev V.Yu., Kryuchkov A.A. VIEW ON MATHEMATICAL COMPUTATIONS BY MEANS OF DATABASES ON EXAMPLE OF MULTIPLICATION AND TRANSPOSE OF MATRICES

The representation of matrices in columnar databases is considered. Realization of the transpose and matrix multiplication by means of databases is shown.

Key words: mathematical computations; matrix multiplication; transposition of matrices; databases; stored routines.