Модел. и анализ информ. систем. Т. **18**, № **3** (2011) 144–154

УДК 519.677

Оптимизация конъюнктов условий в составе запросов

Кузнецов С. Д., Мендкович Н. А.

Институт Системного Программирования РАН

e-mail: mend@f-group.ru

получена 20 июня 2011

Ключевые слова: оптимизация запросов, лексическая оптимизация.

Предлагается новый алгоритм оптимизации запроса. Этот алгоритм сокращает ограничения запросов, содержащих многоатрибутные условия. Он также решает проблему выражений «Условие `AND` Конъюнкция условий».

1. Введение

Несмотря на большое число разработанных методов оптимизации запросов, т.е. изменения запросов с целью ускорения их обработки, эта задача все еще остается актуальной для существующих СУБД. Как отмечает известный специалист в данной области С. Чаудхари, «оптимизация запросов столь же, если не более, актуальна, чем в прошлом. Современные базы данных включают в себя системы обработки транзакций в реальном времени, ERP-системы, системы управления взаимодействием с клиентами, аналитическую обработку в реальном времени, анализ данных с использованием хранилищ данных (Data-warehouses). Запросы, генерируемые этими приложениями, все сложнее, а базы данных – больше чем когда бы то ни было. Таким образом, центральная роль оптимизации запросов, которая ищет различные стратегии обработки и выбирает лучший план исполнения, остается несомненной» [1, p. 961].

На текущий момент разработана масса подходов к решению задачи оптимизации запросов, перечисление которых выходит далеко за пределы данной статьи. В число обзорных работ, посвященных этому предмету, входят [2, 3, 4]. В данной статье мы обращаемся к проблеме работы с вложенными подзапросами, которая является, в какой-то мере, классической, и способы ее решения предлагаются во множестве различных работ. (В частности, решение рассматривается в [5, 6]. Обзоры работ, посвященных этой проблеме, содержатся в [3, раздел 4.2.2; 7, с. 682–684; 8, раздел 2].)

В нашей статье мы обращаемся к проблеме сокращения числа вложенных подзапросов путем объединения семантически близких и удаления избыточных. Алгоритмы, решающие эту задачу, включены в многие широко распространенные СУБД.

Например, в системе Oracle [9, раздел 2] подобная процедура производится над подзапросами на основании «свойства включения» («блок запроса X включает другой блок запроса Y, если результат Y является подмножеством (не обязательно собственным) результата X» [9, p. 1368]).

Однако объединение множества подзапросов в один делает возможным формирование сложной ограничительной части, включающей множество условий, часть из которых может дублировать друг друга. С целью повышения скорости обработки подобных запросов желательно применение алгоритмов, удаляющих или объединяющих избыточные условия.

В наших предыдущих работах мы описали ряд алгоритмов, позволяющих объединить избыточные условия, представимые в виде неравенств с общей левой частью [10, 11]. Цель данных алгоритмов оптимизации – максимальное сокращение числа переменных и повышение лаконичности записи системы неравенств. Мы предложили ряд методов, позволяющих оптимизировать запрос, используя алгоритмы алгебры логики и линейной алгебры, в которых ограничения запросов рассматривались бы как логические функции:

- алгоритм поглощения условий-дубликатов и их конъюнктов, основанный на распознавании дубликатов не только в рамках одного конъюнкта, как в ранее упомянутых работах, но и разных конъюнктах одного ограничения, представленного в дизъюнктивной нормальной форме;
- алгоритм минимизации запроса на основе дополненного алгоритма Квайна [12] (предложения Маккласки [13] нереализуемы для данного вида задач), при котором ограничение в ДНФ рассматривается как множество утверждений.

Однако возможны и более сложные ситуации, когда речь не идет о простом дублировании условий с общей левой частью. Например, левая часть условия включает в себя множество атрибутов, часть из которых являются общими с другим условием (см. пример 1).

Пусть системе задан запрос на список заказов за 7 июля 2008 года, имеющих определенные ограничения по цене и издержкам.

Пример 1.

```
SELECT o_orderpriority,
FROM orders
WHERE o_orderdate => '2008-07-07' AND
  EXISTS (SELECT *
          FROM lineitem
          WHERE l_orderprice < 1000) OR ...
EXISTS (SELECT *
          FROM lineitem
          WHERE l_orderprice+cost < 500 AND l_cost>10
```

После сращивания ограничение данного запроса будет иметь вид:

```
...WHERE orderdate >= '2008-07-07' AND
     EXISTS (SELECT *
          FROM lineitem
(1)          WHERE lorderprice < 1000 OR
(2)          (l_orderprice+l_cost < 500 AND l_cost>10))...
```

В данном случае очевидно, что при обработке запроса в его текущем виде будут выполнены избыточные операции, так как все данные, выбранные в соответствии с условием (2), будут соответствовать и условию (1), так как условия конъюнкции не могут быть реализованы при «`l_orderprice => 1000`».

Установить это несложно, если в соответствии с правилами алгебры логики представить дизъюнкцию

```
(3)   l_orderprice < 1000 OR (l_orderprice+cost < 500 AND l_cost>10)) =
```

следующим образом:

```
(4)   l_orderprice < 1000 OR (l_orderprice => 1000 AND
      l_orderprice+cost < 500 AND l_cost>10)),
```

так как если то или иное условие, стоящее в дизъюнкции, неверно, то верно обратное ему условие:

$$A\,\mathtt{OR}\,B = A\,\mathtt{OR}\,(A\,\mathtt{AND}\,B)\,\mathtt{OR}\,(\,\mathtt{NOT}\,(A)\,\mathtt{AND}\,B) = A\,\mathtt{OR}\,(\,\mathtt{NOT}\,(A)\,\mathtt{AND}\,B).$$

Рассмотрев преобразованное для примера 1 условие (4), мы видим, что конъюнкт становится невыполнимым, поэтому его выполнение не несет смысла и он может быть удален без ущерба для семантики запроса. Оптимальный вид ограничения будет следующим.

```
...WHERE orderdate => '2008-07-07' AND
   EXISTS (SELECT *
           FROM lineitem
WHERE l_orderprice < 1000)...
```

Однако, чтобы провести подобное сокращение запроса, необходим специальный алгоритм, предназначенный для обработки конъюнктов условий, часть из которых содержит более одного атрибута. Некоторые из условий конъюнкта могут быть избыточными или взаимоисключающими, что в первом случае требует их удаления, во втором — удаления самого вложенного подзапроса.

Подчеркнем, что возникновение подобного рода избыточности возможно не только при автоматической генерации запроса, но и в результате его составления человеком-программистом, так как, напомним, речь идет об объединении нескольких вложенных подзапросов, изначально предназначенных для отдельного независимого выполнения. Кроме того, конъюнкты представимых в виде линейных неравенств условий могут быть намеренно сформулированы и на этапе генерации запроса в случае, например, если он ориентирован на проверку эконометрических моделей.

Так или иначе, эти условия могут встречаться и требуют анализа и оптимизации, алгоритм которой мы предлагаем в этой статье.

В [10, p. 189–191] мы кратко охарактеризовали рассматриваемый метод, который помог бы решить проблему обнаружения избыточности в подобных ситуациях. В данной работе мы предлагаем его более подробное описание и ряд дополнений, расширяющих возможности по обработке запросов.

2. Алгоритм

Рассмотрим более сложное ограничение, содержащие в себе большее число условий, в которых задействованы общие атрибуты. В примере 2 представлен запрос на получение имен рабочих, имеющих описанные в ограничении соотношения, и размеров зарплаты, платежей и бонусов.

Пример 2.

```
SELECT workers_name
FROM workers
WHERE (Salary+Bonus>10000
       AND Salary-Payment>3000
       AND 2*Payment>8000)
       OR Payment-Bonus=>100
```

Чтобы проверить конъюнкцию из первых трех условий на избыточность, ее необходимо дополнить *внешним* условием `NOT(Payment-Bonus=>100)=Payment-Bonus<100`. Подчеркнем, что создание этого условия носит временный характер, если его использование не приведет к обнаружению логической избыточности или противоречий в конъюнкте, то оно будет исключено и не будет использоваться в конечном представлении запроса.

Здесь нам придется опуститься на уровень отдельных условий, которые можно представить как алгебраические неравенства. Естественно, тогда любой такой конъюнкт можно рассматривать как систему неравенств, которая описывает некоторую совокупность значений входящих в них переменных.

Смысл описываемого ниже алгоритма оптимизации заключается в том, чтобы исключить условия, семантически повторяющие друг друга, либо противоречащие друг другу и приводящие к невыполнимости конъюнкта, что позволяет исключить его из ограничения и упростить обработку запроса.

Если представить конъюнкт в виде системы неравенств, то мы ищем множество неравенств, исключающих друг друга и делающих систему невыполнимой, или множество базисных неравенств системы (неравенств, входящих в изначальную систему, значение совокупности которых тождественно значению системы).

В случае обнаружения взаимоисключающих неравенств, мы можем избежать дальнейшей обработки данного конъюнкта как заведомо невыполнимого. Мы можем также определить множество базисных неравенств, т.е. подмножество неравенств, выражающих семантику всего конъюнкта, что делает возможным его замену меньшим множеством условий.

В разделе 2.1. данной главы мы опишем пошаговую работу предлагаемого алгоритма, в разделе 2.2. проиллюстрируем ее, показав процесс оптимизации ограничения из примера 2.

2.1. Описание алгоритма

В таком случае для конъюнкта составляется таблица, каждой строке которой соответствует условие, каждому столбцу, кроме двух последних, — переменная (A_{ij}), предпоследнему столбцу — знак операции сравнения, последнему столбцу — свободный член (B_i). В табл. 1 показано внутреннее представление конъюнкта из ограничения из примера 2.

Таблица 1. Внутреннее представление запроса из примера 2

Salary	Bonus	Payment	comp-op	B
1	1	0	$>$	10000
1	0	-1	$>$	3000
0	-1	1	$<$	100
0	0	2	$>$	8000

Мы используем два преобразования, не меняющих решения системы при их применении, без учета того, идет ли речь об уравнениях или неравенствах:

1. умножение выражения на произвольное число;

2. операцию определения следствия из двух неравенств, т.е. формулирования третьего неравенства, являющегося логическим следствием из двух анализируемых.

Предлагаемый нами алгоритм состоит из следующих 6 шагов.

1. Из множества строк выбирается одна строка, содержащая наибольшее число ненулевых значений A_{ij} (далее назовем ее *рабочей*). Если строк с тем же числом ненулевых значений несколько, то из них выбирается одна произвольная. Затем выбираются из прочего множества строк те, которые содержат тот же набор переменных или часть из них.

В случае, если ни одну такую строку подобрать невозможно, мы повторяем процедуру с выбором другой строки в качестве рабочей из множества оставшихся, выбирая те, где больше всего ненулевых переменных.

Затем строки, включая рабочую, копируются в отдельную таблицу, где с ними производятся преобразования, описанные ниже. Если число строк меньше, чем число переменных, то в процессе обработки системы группа переменных, встречающихся только в одной строке, обрабатывается как единая переменная. Например, если в полученной подсистеме есть лишь два неравенства:

$$a + b + c > c_1, \quad a < c_2;$$

где a, b, c — переменные, '+' — знак любой операции, '>' — любой знак сравнения, а c_2, c_1 — константы, $b + c$ обрабатывается как одна переменная.

2. Затем необходимо произвести сопоставление рабочего неравенства с другими неравенствами подсистемы. Для этого для каждой пары элементов A_{ik} и A_{jk}, где

i и j — номера рабочей и парной строк, вычисляем *индекс*, на который их необходимо умножить, чтобы при сложении строк элемент, получившийся в итоге, был равен нулю. Индекс легко вычислить для любой пары чисел с помощью решения следующего уравнения:

$/A_{ik}/-/A_{jk}/*\mathtt{X}=0$, где `X` — неизвестный индекс.

В случае если обе строки содержат ненулевые значения A в одних и тех же полях и для них всех можно вычислить одинаковый индекс, а операция сравнения также одинакова, то одна из строк удаляется как семантически избыточная.

Если парная строка содержит меньшее число ненулевых значений A, чем рабочая, однако для всех A_{jk} неравных 0 — A_{ik} также не равны 0, причем для всех ненулевых A_{jk} можно вычислить одинаковый индекс, сопоставление строк проводится, как это описано в пункте 3.

Если только для части ненулевых пар A_{jk} — A_{ik} индекс является общим, то следствие формулируется с помощью правил из пункта 4.

Если вычислить общий индекс для элементов A не представляется возможным, то выражение с большими элементами преобразуется описанным ниже образом. Так, исходный вид пары

$$n * a + n_1 * b + n_2 * c + n_4 * d > c_1 \quad m * a + m_1 * b + m_2 * c > c_2,$$

где первое неравенство рабочее, второе — парное, n и m — константы. Чтобы преобразовать выражение, следует выделить общий атрибут из строки, где общее число атрибутов больше. Тогда если $n > m$, мы определяем атрибут как выражение $n * a = m * a + (n - m) * a$, если верно обратное неравенство, то
$n * a = m * a - (m - n) * a$.

Если индекс первого выражения больше второго, то выражение преобразуется к следующему виду: $m*a+m_1*b+m_2*c+(n-m)*a+(n_1-m_1)*b+(n_2-m_2)*c+n_4*d > c_1$ $m * a + m_1 * b + m_2 * c > c_2$, где $m * a + m_1 * b + m_2 * c$ становится общим множеством элементов левой части, благодаря чему в соответствии с пунктом 3 относительно этой пары неравенств может быть определено следствие.

Таблица 2. Общий вид определения следствий для пар неравенств, где левая часть парного полностью входит в рабочее неравенство

	$a + b = c_1$	$a - b = c_1$	$a + b > c_1$	$a + b < c_1$	$a - b > c_1$	$a - b < c_1$
$b > c_2$	$a < c_1 - c_2$	$a > c_2 + c_1$	–	$a < c_1 - c_2$	$a > c_1 + c_2$	–
$b < c_2$	$a > c_1 - c_2$	$a < c_2 + c_1$	$a > c_1 - c_2$	–	–	$a < c_1 + c_2$
$a > c_2$	$b < c_1 - c_2$	$b > c_2 - c_1$	–	$b < c_1 - c_2$	–	$b > c_2 - c_1$
$b < c_2$	$b > c_1 - c_2$	$b < c_2 - c_1$	$b > c_1 - c_2$	–	$b < c_2 - c_1$	–

3. В данном случае рабочее неравенство, которое сравнивается с парным, должно быть суммой двух элементов: неизвестного и содержащегося в левой части парного неравенства. Если из рабочего и парного неравенств можно вывести значимое для оптимизации следствие, согласно таблице 2 (см. выше), то индекс умножается на второе (парное) неравенство, после этого второе неравенство сопоставляется с рабочим и на их основе определяется следствие, т.е. формулируется неравенство,

касающееся переменной или группы переменных рабочего неравенства, отсутствующих в парном. В общем виде исходная пара неравенств выглядит так: $a + b > c_1$, $a > c_2$, где a и b — переменные или их группы, c_2 и c_1 — константы. Правила формулирования следствия представлены в виде таблицы 2. В ней первый столбец соответствует парному неравенству, первая строка — рабочему, а все прочие клетки общему виду следствия для конкретного случая. Прочерк означает, что на основе стандартного алгоритма в данном случае невозможно вывести значимое для оптимизации следствие.

При равенстве $a = const$ переменная a заменяется на константу из правой части во всех условиях, поэтому не учитывается в представленной выше таблице.

В общем случае неравенство, с которым производится сравнение, можно описать как сумму двух элементов: неизвестного и содержащегося в левой части второго неравенства. Процедура сопоставления рабочего неравенства с парными производится, пока не перебраны все прочие или пока в рабочем неравенстве остается только одна переменная.

4. В этом случае рабочее и парное неравенство содержат одинаковую переменную или группу переменных. При этом пара неравенств имеет следующий вид: $a + b > c_1$, $a + c > c_2$, где a, b, c — переменные или их группы, '+' — знак любой операции, '>' — любой знак сравнения, а c_2, c_1 — константы. Цель алгоритма — сформулировать следствие, касающееся двух не совпадающих переменных (или их групп) b и c. Правила формулирования следствий для таких случаев описаны ниже в таблице 3.

Таблица 3. Общий вид определения следствий для пар неравенств, где часть элементов левых частей обоих неравенств совпадают

	$a + b > c_1$	$a + b < c_1$	$a - b > c_1$	$a - b < c_1$
$a + c > c_2$	–	$c - b > c_2 - c_1$	–	$b + c > c_2 - c_1$
$a + c < c_2$	$c - b < c_2 - c_1$	–	$b + c < c_2 - c_1$	–
$a - c > c_2$	–	$b + c < c_1 - c_2$	–	$b - c > c_2 - c_1$
$a - c < c_2$	$b - c > c_1 - c_2$	–	$b - c < c_2 - c_1$	–
$c - a < c_2$	–	$b + c < c_2 + c_1$	–	$c - b < c_2 + c_1$
$c - a > c_2$	$b + c > c_2 + c_1$	–	$c - b > c_2 + c_1$	–
$c - b > c_2$	$a + c > c_2 + c_1$	–	–	$c - a > c_2 - c_1$
$c - b < c_2$	–	$a + c < c_2 + c_1$	$c - a > c_2 - c_1$	–
$b - c > c_2$	–	$a + c < c_1 - c_2$	$a - c > c_2 + c_1$	–
$b - c > c_2$	$a + c > c_1 - c_2$	–	–	$a - c < c_2 + c_1$

После определения следствий для всех пар элементов подсистемы процедура поиска парных неравенств и определения следствий применяется для всех неравенств подмножества, содержащих более одной переменной, причем каждому из них поочередно присваивается статус рабочего, т.е. прочие неравенства подсистемы сравниваются с ним как потенциальные парные. Целью шагов алгоритма 3 и 4 является создание всех возможных следствий из условий конъюнкта. Их применение описывается ниже.

5. На этом шаге происходит собственно поиск противоречивых и избыточных условий. В первом случае для каждого из неравенств проводится простой поиск условий или следствий с той же левой частью. В случае, если пара условий и/или оказываются взаимоисключающими (т.е. их конъюнкция равна пустому множеству), то весь конъюнкт исключается из обработки. (Следует применять полученные выводы сразу после обработки каждой подсистемы, так как есть возможность обнаружить логическое противоречие и закончить процесс обработки данной группы условий). С целью поиска избыточных условий для каждого из неравенств генерируется пара условий, обратных проверяемому. Это либо пара строгих неравенств, либо равенство и обратное строгое неравенство, т.е. два ограничения с теми же левой и правой частью, но другими операторами сравнения. Оба эти условия поочередно объединяются с множеством отобранных в подсистемы, в которых формулируются новые выводы и производится поиск противоречий описанным ранее способом. Если проверка одного из вариантов не обнаруживает логических противоречий, то проверка прекращается и условие признается неизбыточным. Если логические противоречия найдены в обеих подсистемах, выражение избыточно. Опишем процедуру анализа на следующем примере (пример 3):

Пример 3.

```
... WHERE a+b>10 AND a>10 AND b>0 AND c>8
```

Для условия $a + b > 10$ (рабочая строка) отбирается два парных условия `a>10 AND b>0`. На их основе нельзя сформулировать ни одного следствия. Создание других подсистем в рамках конъюнкта также невозможно. Поиск противоречий не дает результатов. В целях поиска избыточности формируются два конъюнкта и следствия к ним, представленные в таблице 4.

Таблица 4. Подсистемы и следствия из них для примера 3

Подсистемы	Следствия
a+b<10 AND a>10 AND b>0	$a < 10$ и $b < 10$
a+b=10 AND a>10 AND b>0	$b < 0$ и $a < 10$

В обоих случаях выводы противоречат существующим условиям. Вывод: условие $a + b > 10$ и конъюнкт $a > 10$ `AND` $b > 0$ описывают одно и то же множество данных. Следовательно, конъюнкт $a > 10$ `AND` $b > 0$ избыточен и подлежит удалению из итогового ограничения. Если исходное условие сопоставлялось не только с изначальными условиями, но и выводами, то после удаления условия использованные выводы сохраняются в конечном представлении.

Итоговый вид условия из примера 3:

```
...WHERE a>10 AND b>0 AND c>8
```

6. В случае, если конъюнкт не был определен как противоречивый, то на данном этапе из него исключаются все созданные в процессе работы следствия, а также внешнее условие, если оно было создано перед обработкой конъюнкта.

Описанный подход позволит создать максимальное множество элементарных выводов с целью выявить возможные логические противоречия в ограничении или

избыточные условия, упростив таким образом обработку запроса, отказавшись от выполнения излишних условий или невыполнимых запросов и подзапросов.

2.2. Пример оптимизации ограничения запроса

Теперь опишем пошаговое выполнение алгоритма для примера 2. При выборе рабочих неравенств мы сталкиваемся с тремя возможными кандидатами для разных подсистем. Это первые три условия, содержащие по 2 переменных. Условие `Salary+Bonus>10000` отвергается, так как не удается найти другие неравенства, содержащие только элементы, входящие в него. Условия `Payment-Bonus<100` и `Salary-Payment>3000` позволяют создать две подсистемы. Первая из них:

```
Salary-Payment>3000 AND
2*Payment>8000.
```

С помощью приведенного алгоритма на их основе формулируются следствия:

```
Payment>4000 AND
Salary>7000.
```

Вторая:

```
Payment-Bonus<100 AND
2*Payment>8000.
```

Из данной системы невозможно вывести следствия, значимые для оптимизации. Из пары условий `Salary+Bonus>10000 AND Salary-Payment>3000` невозможно вывести никаких значимых для оптимизации следствий. Для `Salary-Payment>3000 AND Payment-Bonus<100` также невозможно сделать значимые выводы. На основе полученных следствий `Payment>4000 AND Salary>7000` и выражения `Salary+Bonus>10000` делается вывод: `Bonus<=3000`. Из условия `Payment-Bonus<100` и следствия `Bonus<=3000`, выводится следствие: `Payment<3100` (вернее два одинаковых следствия), противоречащее следствию `Payment>4000`. Действительно, разность платежа, превышающего 4000, и бонуса, меньшего или равного 3000, не может быть меньше 100. Следовательно, ограничение из примера 2 будет иметь оптимальную форму `Payment-Bonus=>100`.

3. Заключение

Предложенный алгоритм дает возможность более эффективной оптимизации групп условий, объединенных конъюнкцией, а также оптимизации дизъюнкций вида «Условие `OR` Конъюнкт условий». Последняя задача становится тем более актуальной в контексте реализации алгоритмов слияния вложенных подзапросов в системе Oracle (см. [9]), так как именно при объединении изначально разделенных подзапросов наиболее вероятно обнаружение избыточностей и логических противоречий, обнаружение и исключение каковых значительно упрощает обработку запроса.

Создание новых условий в данном алгоритме не оказывает значимого влияния на затраты при реализации запроса, так как эти новые условия не подвергаются обработке и автоматически исключаются из запроса перед завершением процесса оптимизации и исполнением запроса.

Список литературы

1. Chaudhuri S. Query Optimizers: Time to Rethink the Contract? // Proceedings of the ACM SIGMOD International Conference on Management of Data, Providence, Rhode Island, USA, June 29 - July 2, 2009.

2. Ionnidis Y. E. Query Optimization // The Computer Science and Engineering Handbook. Boca Raton, CRC Press, 1996.

3. Chaudhari S. An Overview of Query Optimization in Relational Systems // Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington. ACM Press, 1998. Русский перевод: Чаудхари С. Методы оптимизации запросов в реляционных системах // СУБД. 1998. №3.

4. Кузнецов С. Д. Методы оптимизации выполнения запросов в реляционных СУБД. `http://www.citforum.ru/database/articles/art_26.shtml`. Доступ 10 июня 2011 года.

5. Muralikrishna M. Improved Unnesting Algorithms for Join Aggregate SQL Queries // Proceedings of the 18th International Conference on Very Large Data Bases, August 23-27, Vancouver, Canada, 1992.

6. Khaitan P., Satish K. M., Korra S. B., Jena S. K. Improved Query Plans for Unnesting Nested SQL Queries // Proceedings of 2nd International Conference on Computer Science and its Applications, December 10-12, South Korea, 2009.

7. Дейт К. Дж. Введение в системы баз данных. Москва; Санкт-Петербург; Киев, 2001.

8. May N., Helmer S., Moerkotte G. Strategies for Query Unnesting in XML Databases // ACM Transactions on Database Systems. 2006. №331(3).

9. Bellamkonda S., Ahmed R., Witkowski A., Amor A., Zait M., Lin C.-C. Enhanced Subquery Optimizations in Oracle // Proceedings of the 35th international conference on Very large data base, Lyon, France, August 28, 2009.

10. Mendkovich N., Kuznetcov S. New Algorithms for Lexical Query Optimization // Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces. June 22-25, 2009, Cavtat/Dubrovnik, Croatia. Edited by Vesna Luzar-Stiffler, Iva Jarec, Zoran Bekic. Technical Editor Boris Grinfeld.

11. Кузнецов С. Д., Мендкович Н. А. Новые алгоритмы лексической оптимизации запросов // Моделирование и анализ информационных систем. 2009. Т. 16, №4.

12. Quin W. V. O cores and prime implicants of truth functions // American Mathematics Monthly. 1959. V. 66. №9.

13. McCluskey E. J. Minimization of Boolean Functions // The Bell System Technical Journal. November 1956. V. 35, Issue 5.

Optimization of Queries Containing Conjunctions of Conditions

Kuznetsov S. D., Mendkovich N. A.

Keywords: query optimization, lexical optimization.

A new algorithm for query optimization is proposed. This algorithm simplifies queries restriction containing multi-attribute conditions. It also solves the problem of "Condition AND Conjunction of conditions" expressions.

Сведения об авторах:
Сергей Дмитриевич Кузнецов,
Институт Системного Программирования РАН,
профессор, главный научный сотрудник;
Никита Андреевич Мендкович,
ООО «Объединение сетей ФРИнет», инженер.