Организация баз данных
2003. № 1(5)

УДК 004.65

© 2003 г. **Н.А. Гребенников**,
Ю.А. Григорьев, д-р техн. наук
(Московский государственный технический университет им. Н.Э.Баумана)

МЕТОДЫ ПОИСКА ОПТИМАЛЬНОГО ПЛАНА НА ОСНОВЕ НИСХОДЯЩЕЙ СТРАТЕГИИ

Рассматриваются новые методы поиска оптимального плана в СУБД на основе нисходящей стратегии анализа альтернативных планов. Доказывается полнота и качество предлагаемых методов, оценивается эффективность их использования на основе анализа вероятностей отсечения несвязанных групп в пространстве поиска.

Введение

Системы управления базами данных (СУБД) являются основным компонентом современных систем обработки информации. С целью уменьшения времени выполнения запросов в состав СУБД включается оптимизатор запросов, задачей которого является анализ множества альтернативных планов выполнения запросов и выбор оптимального плана. Основным компонентом оптимизатора запросов СУБД является модуль поиска оптимального плана.

Раньше в оптимизаторах запросов использовалась восходящая стратегия поиска, основанная на методе динамического программирования и впервые реализованная в оптимизаторе системы System R [1]. Оптимизаторы, построенные на основе этого подхода, имеют ограниченную расширяемость. Добавление нового оператора, такого как функция агрегации, требует множества изменений в оптимизаторе. Кроме того, указанный алгоритм непригоден для запросов, включающих соединение более 15 таблиц, так как объем необходимой памяти для работы алгоритма динамического программирования имеет экспоненциальную зависимость от числа соединяемых таблиц. Позднее были предложены два пути для построения расширяемых оптимизаторов. Лохман [2] предложил использовать правила для генерации планов в восходящем оптимизаторе, а Грэф и Де Витт [3] – трансформации для генерации новых планов с использованием нисходящего подхода.

Порождающие правила Лохмана были реализованы в восходящем расширяемом оптимизаторе Starburst [4]. Использование оптимизатора

Starburst показало возможность его применения в различных случаях: от возрастающих соединений [5] до распределенных гетерогенных баз данных [6]. Однако оптимизатор Starburst использовал восходящий подход динамического программирования System R, который генерирует $O(3^N)$ различных планов [7]. Из-за экспоненциального роста числа планов оптимизатор Starburst использовал эвристики на фазе преобразований при оптимизации сложных запросов [8], что, как показано в работе [9], достаточно часто приводило к выбору неоптимального плана.

Первым оптимизатором, использующим нисходящую стратегию поиска, был оптимизатор Exodus [3]. Основной целью этого проекта была демонстрация расширяемости Exodus. Позднее Грэф разработал оптимизатор Volcano [10], его целью было повышение производительности СУБД за счет использования алгоритма «запоминания». Но производительность оптимизатора Volcano была невысокой из-за применяемой стратегии поиска, при которой все логические выражения генерировались перед генерацией физических выражений. Поэтому Volcano генерировал $O(3^N)$ выражений – это столько же, сколько и в оптимизаторе Starburst.

В середине 90-х гг. появилось новое поколение оптимизаторов, использующих методы объектно-ориентированного программирования, что значительно облегчило задачу построения или расширения оптимизатора. Примерами таких оптимизаторов являются системы OPT++ [11] и Cascades [12]. Система OPT++ позволяет сравнивать производительность нисходящих и восходящих оптимизаторов, однако в нисходящем случае она использует стратегию поиска оптимизатора Volcano и поэтому показывает низкую производительность. Оптимизатор Cascades был разработан с целью демонстрации возможностей расширяемости объектно-ориентированного подхода и высокой производительности нисходящего оптимизатора, которая достигалась за счет использования механизма стоимостного отсечения групп, недоступного при восходящем подходе. Нисходящий расширяемый подход Cascades наиболее перспективный из рассмотренных выше методов, коммерческие СУБД от Microsoft [13] и Tandem [14] основываются на оптимизаторе Cascades. Однако в существующей реализации этот оптимизатор показывает все же недостаточно высокую производительность, так как вероятность срабатывания механизма отсечения групп невелика, что снижает эффективность его использования.

В статье предложены новые методы поиска оптимального плана на основе нисходящей стратегии, устраняющие указанные недостатки.

Основные определения

Алгоритм нисходящей стратегии поиска плана, используемый в оптимизаторе Cascades, основывается на применении специального внутреннего представления пространства поиска. Для описания этого представле-

ния введем ряд определений. *Логический оператор* – функция, отображающая входы оператора на его выходы. *Физический оператор* – это алгоритм, позволяющий получить выходные кортежи на основе кортежей его входов. *Операторное выражение* (*выражение*) – дерево операторов, в котором операторы с нулевым числом входов выступают в качестве листьевых узлов дерева (они определяют доступ к хранимому отношению), а операторы с ненулевым числом входов – в качестве внутренних узлов. При этом входы каждого оператора являются выходами его дочерних операторов. Выражение называется логическим или физическим в зависимости от типа оператора верхнего уровня. Начиная с оптимизатора Volcano, пространство поиска в нисходящих оптимизаторах обозначается термином *memo-структура* [10]. Memo-структура состоит из двух взаимно рекурсивных структур данных, которые называются мультивыражениями и группами. *Мультивыражение* – это оператор, входами которого являются группы (примером мультивыражения является соединение двух групп отношений). По определению все иерархически связанные выражения с одинаковым оператором верхнего уровня и одинаковыми входами этого оператора представляются одним мультивыражением. *Группа* – класс эквивалентных мультивыражений, позволяющих получить один и тот же результат (т.е. одинаковые выходные наборы кортежей).

Анализ механизма отсечения групп

Как указано в работе [12], основным преимуществом нисходящего оптимизатора запросов над восходящим является возможность использовать механизм отсечения групп при оптимизации запроса. Отсечение группы – ситуация, при которой группа не будет оптимизирована, т.е. для неё не будет сгенерировано ни одно мультивыражение в процессе оптимизации. Отсечение группы является пассивным действием, т.е. группа никогда вручную не удаляется. Отсеченная группа будет содержать только одно мультивыражение, которое было использовано для ее инициализации.

Отсечение групп приводит к уменьшению числа мультивыражений, рассматриваемых в процессе оптимизации (сокращению пространства поиска) и, следовательно, к уменьшению времени оптимизации. При этом оптимизатор, который осуществляет отсечение групп, гарантированно строит оптимальные планы (в отличие от методов сокращения пространства поиска, основанных на эвристиках). Количественный выигрыш от использования механизма отсечения групп зависит от частоты срабатывания отсечения. Чем чаще происходит отсечение групп, тем больше сокращается пространство поиска и тем меньше время оптимизации. Таким образом, представляется целесообразным разработка методов, позволяющих увеличивать вероятность отсечения групп в процессе оптимизации.

Рассмотрим процесс вычисления стоимостей при оптимизации груп-

пы в соответствии с алгоритмом нисходящей стратегии поиска [12]. На рис. 1 показан псевдокод рекурсивного алгоритма вычисления стоимостей в процессе оптимизации.

```
(1)  Cопт(k, группа, CВГ)  (сначала для корневой группы CВГ = ∞)
(2)  Цикл по физическим мультивыражениям группы (по i)
{
(3)       CНГ(i) = стоимость базового оператора i-го физического мультивыражения.
(4)       Если CНГ(i)>CВГ, то переход к i+1
(5)       Цикл по входным группам j физического мультивыражения i
          {
(6)            Получение стоимости оптимально плана j-го входа: Cопт = Cопт(i, вход j, CВГ – CНГ(i)).
(7)            Если Cопт =-1 (оптимального плана не существует), то переход к i+1.
(8)            CНГ(i) = CНГ(i) + Cопт.
          }
(9)       Если CНГ(i)<CВГ , то CВГ = CНГ(i).
}
```

Рис. 1. Стоимостный анализ алгоритма нисходящей стратегии поиска.

На рис. 2 графически изображен процесс изменения стоимостей в процессе оптимизации группы, затрагивающий N таблиц запроса соединения.

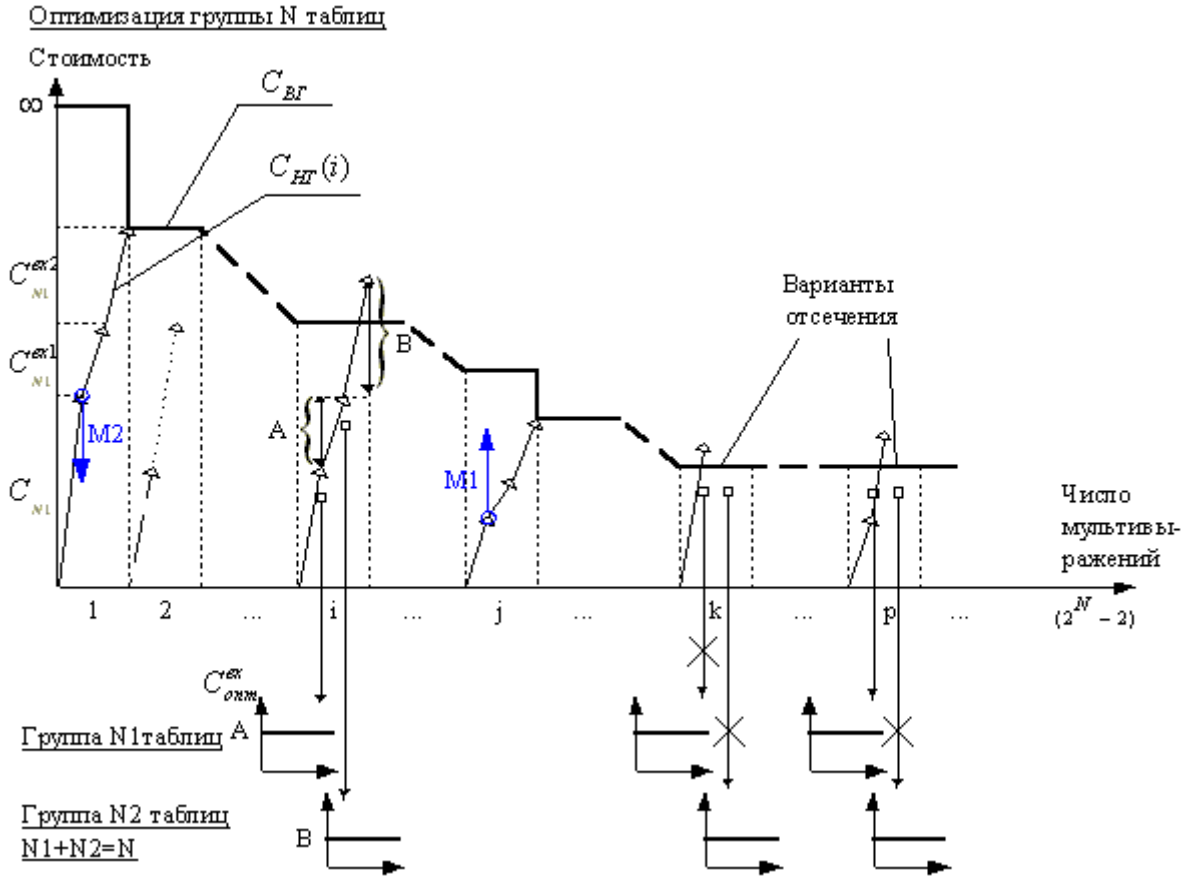

Рис. 2. Изменение стоимостей при оптимизации группы N таблиц.

Вычисление стоимостей в процессе оптимизации группы происходит следующим образом.

Вычисляется стоимость базового оператора текущего физического

мультивыражения группы (см. (3) на рис. 1).

Производится сравнение этой стоимости с текущим значением верхней границы стоимости группы (см. (4) на рис. 1).

Если данная стоимость превышает верхнюю границу, то выполняется переход к следующему мультивыражению группы (см. цикл (2) на рис. 1).

Вычисляются стоимости оптимальных планов первой и второй входных групп текущего физического мультивыражения путем оптимизации этих групп (см. цикл (5) и строку (6) на рис. 1).

Определяется стоимость текущего физического мультивыражения группы как сумма трех указанных стоимостей: стоимости базового оператора (см. (3) на рис. 1) и стоимостей оптимальных планов первой и второй входных групп (см. (8) на рис. 1).

Производится сравнение стоимости текущего физического мультивыражения с текущим значением верхней границы стоимости группы (см. (9) на рис. 1).

Если эта стоимость меньше верхней границы, то верхняя граница стоимости группы устанавливается равной стоимости текущего физического мультивыражения,

Шаги с 1-го по 7-й повторяются для всех оставшихся мультивыражений исходной группы.

Полученная таким образом верхняя граница стоимости группы представляет собой стоимость оптимального мультивыражения группы и может быть использована при оптимизации групп, чьи мультивыражения содержат данную группу в качестве входов (на шаге 4).

Таким образом, стоимость каждого мультивыражения в группе определяется в три этапа:

вычисляется стоимость базового оператора мультивыражения;

добавляется стоимость оптимального плана первого входа мультивыражения;

добавляется стоимость оптимального плана второго входа мультивыражения.

На рис. 2 приведены возможные ситуации, возникающие при вычислении и сравнении стоимостей в процессе оптимизации.

Мультивыражение i: конечная стоимость мультивыражения превышает текущую верхнюю границу стоимости группы, поэтому значение верхней границы стоимости группы на этой итерации не изменяется.

Мультивыражение j: конечная стоимость мультивыражения меньше текущей верхней границы стоимости группы, поэтому значение верхней границы стоимости группы устанавливается равным стоимости мультивыражения.

Мультивыражение k: стоимость базового оператора мультивыражения превышает верхнюю границу стоимости группы, поэтому оптимиза-

ции входных групп этого мультивыражения не происходит (отсечение групп входов).

Мультивыражение p: сумма стоимостей базового оператора и оптимального плана первого входа мультивыражения превышает верхнюю границу стоимости группы, поэтому оптимизации второй входной группы этого мультивыражения не происходит (отсечение группы второго входа).

Как видно из приведенного описания, условием, определяющим возможность отсечения группы, является то, что текущая стоимость рассматриваемого мультивыражения группы (в дальнейшем ее будем обозначать как НГ) превышает текущую верхнюю границу стоимости группы (ВГ). При этом стоимость НГ в момент сравнения включает стоимость базового оператора мультивыражения и, возможно, стоимость оптимального плана первого входа.

Указанное условие будет выполняться тем чаще, чем больше НГ и чем меньше ВГ. Поэтому в настоящей работе предлагаются два метода улучшения работы алгоритма нисходящей стратегии поиска. В первом методе (М1 на рис. 2) предлагается увеличить значение стоимости мультивыражения НГ в момент сравнения (за счет учета информации о входах, которые были оптимизированы ранее, и использования логических свойств для других входов). Второй метод (М2 на рис. 2) определяет метод генерации пространства поиска, при использовании которого первые мультивыражения групп в пространстве поиска будут иметь небольшую стоимость базового оператора и его входов, что позволит достичь раннего уменьшения верхней границы стоимости группы (этот оператор не будет являться декартовым произведением таблиц).

Метод увеличения стоимости мультивыражения

Выше отмечалось, что нисходящая стратегия поиска оптимизатора Cascades может приводить к отсечению групп в двух случаях:

стоимость базового оператора мультивыражения превышает верхнюю границу стоимости группы, поэтому оптимизация входных групп этого мультивыражения не выполняется;

сумма стоимостей базового оператора и оптимального плана первого входа мультивыражения превышает верхнюю границу стоимости группы, поэтому оптимизации второй входной группы этого мультивыражения не происходит.

В данном разделе предлагается метод увеличения значения стоимости мультивыражения в момент сравнения за счет учета информации о входах, которые были оптимизированы ранее, и использования логических свойств для других входов. Увеличение нижней границы будет приводить к более частому срабатыванию механизма отсечения группы. На рис. 3 показаны изменения в алгоритме нисходящей стратегии поиска (см. рис. 1),

реализующие указанный подход.

(3a) $C_{нг}(i)$ = стоимость базового оператора MExpr +
(3б) стоимость входов, которые уже имеют оптимальные планы для требуемых свойств +
(3c) стоимость копирования результатов остальных входов

(5a) Для каждого оптимизируемого входа MExpr, для которого ранее не был найден оптимальный план для требуемых свойств {

(8a) $C_{нг}(i) = C_{нг}(i)$ + стоимость оптимального плана входа - стоимость копирования результатов входа

Рис. 3. Метод увеличения нижней границы стоимости группы (M1).

Приведенные на рис. 3 изменения позволяют избежать оптимизации входных групп мультивыражения (в строке (6) на рис. 1) путем добавления в строках (3б) и (3c) стоимостей тех входных групп, которые могут быть получены без проведения их оптимизации. Если сумма этих стоимостей будет превышать верхнюю границу стоимости группы, то нет необходимости оптимизировать оставшиеся входные группы.

Как показано на рис. 3, предлагается вычислять исходную стоимость мультивыражения как сумму трех компонентов: стоимости базового оператора мультивыражения (строка (3a)), стоимостей входов, которые имеют оптимальные планы для требуемых свойств, т.е. входов, которые уже были оптимизированы ранее (строка (3б)), и стоимостей тех входных групп, которые пока не имеют оптимальных планов (строка (3c)).

Для входных групп, для которых оптимальные планы были получены ранее (при поиске в другом поддереве), могут использоваться уже вычисленные стоимости без повторной оптимизации этих групп. Значение стоимости для какой-либо оставшейся входной группы представляет собой стоимость передачи набора кортежей, являющегося результатом оптимизации мультивыражения группы, к следующему оператору в дереве запроса. Эта стоимость может быть оценена на основе выбранной стоимостной модели и размеров результатов групп без их оптимизации, так как размер результата является логическим свойством группы и одинаков для всех мультивыражений в ней.

Если вычисленная таким образом стоимость мультивыражения превышает текущую верхнюю границу стоимости группы, то оставшиеся входные группы текущего мультивыражения не оптимизируются. В противном случае необходима оптимизация оставшихся входных групп. Для этой цели организуется цикл по всем входным группам текущего мультивыражения, чьи оптимальные планы не были найдены ранее (строка (5a)). В цикле производится поиск оптимального плана входной группы, далее из значения стоимости текущего мультивыражения исключается стоимость копирования результата входной группы, которая была добавлена ранее (строка (3c)), и замещается полной стоимостью оптимального плана этой входной группы, включающей в себя и стоимость копирования результата

(строка (8а)).

Метод генерации пространства поиска без декартовых произведений

В данном разделе предлагается метод генерации пространства поиска, при использовании которого первые мультивыражения большинства групп в этом пространстве будут иметь небольшую стоимость базового оператора, что позволит достичь раннего уменьшения верхней границы стоимости групп.

Пеленкофт показал в работе [15], что для запросов соединения достаточно четырех правил преобразования для генерации всех уникальных логических мультивыражений в группе:

П-1: коммутативность $x \bowtie y \rightarrow y \bowtie x$,

П-2: правая ассоциативность $(x \bowtie y) \bowtie z \rightarrow x \bowtie (y \bowtie z)$,

П-3: левая ассоциативность $x \bowtie (y \bowtie z) \rightarrow (x \bowtie y) \bowtie z$,

П-4: обмен $(w \bowtie x) \bowtie (y \bowtie z) \rightarrow (w \bowtie y) \bowtie (x \bowtie z)$.

В настоящей работе предлагается метод генерации пространства поиска, при котором всегда используется лево-глубокое исходное дерево запроса и применяется набор правил, включающий только два правила – **П-1** и **П-2**. Использование лево-глубокого исходного дерева запроса допустимо, так как любое операторное дерево, состоящее только из операторов соединения и доступа к отношениям, логически эквивалентно лево-глубокому операторному дереву.

При оптимизации предлагается использовать следующий набор правил и условия их применения.

УП-1:

1) мультивыражение $y \bowtie x$ добавляется в группу [xy] с мультивыражением $x \bowtie y$,

2) для $y \bowtie x$ запрещается применение правил УП-1 и УП-2.

УП-2:

1) мультивыражение $x \bowtie (yz)$ добавляется в группу [xyz] с мультивыражением $(xy) \bowtie z$,

2) для $x \bowtie (yz)$ запрещается применение правила УП-2,

3) создается новая группа [yz] с первым мультивыражением $y \bowtie z$,

4) для $y \bowtie z$ разрешается применение правил УП-1 и УП-2,

5) при этом x, y – группы, z – однотабличная группа.

Предлагаемый метод генерации пространства поиска для запроса соединения *k* таблиц состоит из следующих шагов:

создать группу верхнего уровня $[A_1,\ldots, A_k]$, включив в нее одно

мультивыражение $[A_1,\ldots, A_{k-1}]\bowtie[A_k]$;

применить правила УП-1 и УП-2 к данному мультивыражению;

применить правила УП-1 и УП-2 к первым мультивыражениям всех новых групп, созданных на шаге 2 с помощью правила 3) УП-2;

Процесс оптимизации группы соединения 4 таблиц [xyzw] с использованием набора правил УП-1, УП-2 показан ниже:

$[xyz] \bowtie [w] \xrightarrow{-П1} [w] \bowtie [xyz]$

$([x][yz]) \bowtie [w] \xrightarrow{-П2} [x] \bowtie [yzw] \xrightarrow{-П1} [yzw] \bowtie [x]$

$([y][xz]) \bowtie [w] \xrightarrow{-П2} [y] \bowtie [xzw] \xrightarrow{-П1} [xzw] \bowtie [y]$

$([z][xy]) \bowtie [w] \xrightarrow{-П2} [z] \bowtie [xyw] \xrightarrow{-П1} [xyw] \bowtie [z]$

$([xy][z]) \bowtie [w] \xrightarrow{-П2} [xy] \bowtie [zw] \xrightarrow{-П1} [zw] \bowtie [xy]$

$([xz][y]) \bowtie [w] \xrightarrow{-П2} [xz] \bowtie [yw] \xrightarrow{-П1} [yw] \bowtie [xz]$

$([yz][x]) \bowtie [w] \xrightarrow{-П2} [yz] \bowtie [xw] \xrightarrow{-П1} [xw] \bowtie [yz]$

Видно, что суммарное число мультивыражений в группе равно 14, что соответствует общей формуле $2^k - 2$ при $k=4$.

Следующая лемма доказывает, что, когда входное операторное дерево является лево-глубоким, набор правил УП-1, УП-2 позволяет генерировать все возможные эквивалентные логические мультивыражения.

Лемма 1 (полнота набора правил). Пусть R – лево-глубокое операторное дерево. Предположим, что для R применяется предлагаемый в настоящей работе метод генерации пространства поиска (правила УП-1 и УП-2). Тогда справедливы следующие утверждения: (1) каждая оптимизированная группа будет содержать все возможные эквивалентные логические мультивыражения; (2) если группа содержит более одной таблицы, то второй вход первого мультивыражения группы будет однотабличной группой; (3) новые группы будут создаваться только путем применения правила ассоциативности.

Лемма 1 указывает на то, что порядок оптимизации можно изменять так, что первое мультивыражение в каждой группе будет иметь однотабличную группу в качестве правого входа. Однако левый вход в общем случае может включать декартово произведение. Во многих случаях соединения, включающие декартовы произведения, являются наиболее «дорогими». Следующая теорема доказывает, что при использовании предлагаемого метода генерации пространства поиска для связанных ациклических запросов первые мультивыражения всех связанных групп в пространстве поиска будут содержать хотя бы один план, который не включает декартовых произведений.

Связанный запрос – это запрос, граф соединений которого является связным. Группа называется *связанной*, если соответствующий ей запрос (подзапрос) связанный. Любой план, выводимый из несвязанной группы,

будет содержать хотя бы одно *декартово произведение* отношений.

Теорема 1 (качество метода). Пусть Q – это связанный ациклический запрос, и его исходное операторное дерево R лево-глубокое. Пусть для R применяется предлагаемый выше метод генерации пространства поиска (правила УП-1 и УП-2). Тогда каждая связанная группа в результирующем пространстве поиска будет начинаться с мультивыражения, содержащего хотя бы один план без декартовых произведений.

Анализ эффективности разработанных методов поиска оптимального плана

Ниже выполнена оценка выигрыша, получаемого за счет использования метода увеличения стоимости группы М1 и метода генерации пространства поиска М2 при поиске оптимального плана выполнения запроса. Для этого проводится оценка размеров пространства поиска для различных типов деревьев поиска и топологий исходного запроса.

Время поиска оптимального плана выполнения запроса определяется размером пространства поиска, которое в свою очередь определяется числом возможных *деревьев поиска* для исходного запроса. Количество альтернативных деревьев поиска для запросов, содержащих только операторы соединения и доступа к файлам, представляет собой число альтернативных порядков соединения или деревьев соединений.

Дерево соединений – это регулярное бинарное дерево, «листья» которого представляют собой базовые таблицы, указанные в исходном запросе, а промежуточные узлы моделируют операторы соединения. Эти операторы получают входные таблицы через входные дуги и пересылают таблицу результата операции по выходной дуге к следующему оператору. «Корень» (вершина) дерева определяет результат всего запроса. Дерево соединений является бинарным, так как реляционный оператор соединения имеет два входа.

При поиске оптимального плана оптимизатор обычно строит деревья поиска одного из двух типов: лево-глубокие или упорядоченные кустовые. *Лево-глубокое* (левостороннее, левонаправленное, леволинейное, линейное) дерево поиска – это дерево поиска, в котором для каждого оператора соединения (промежуточного узла) хотя бы один из входов является базовой таблицей; в противном случае дерево соединений называется *кустовым*.

Существуют три типовые топологии (графов) запросов: линейная, звездообразная и полностью соединенная.

Линейный (строковый) запрос для *n* отношений состоит из двух терминальных (крайних) отношений, каждое из которых соединено только с одним отношением, и *n*-2 внутренних отношений, каждое из которых соединено с двумя другими соседними отношениями. Связь между двумя

отношениями означает наличие в исходном запросе предиката соединения для этих отношений. *Звездообразный* запрос для *n* отношений состоит из одного центрального отношения и *n*-1 терминальных отношений, которые соединены только с центральным. *Полностью соединенный* запрос для *n* отношений состоит из *n* отношений, каждое из которых соединено со всеми другими *n*-1 отношениями. Любой запрос можно представить в виде полностью соединенного запроса, достроив недостающие связи путем добавления фиктивного предиката соединения, который всегда имеет значение «истина». Добавленные таким образом связи, очевидно, будут представлять декартовы произведения соответствующих отношений.

В процессе оптимизации при исследовании пространства поиска в memo-структуре создаются группы двух типов: связанные и несвязанные. Для количественной оценки выигрыша от использования предлагаемых методов сделаем следующее предположение: использование предлагаемых в настоящей работе методов поиска оптимального плана на основе нисходящей стратегии будет приводить к отсечению всех несвязанных групп в пространстве поиска. Таким образом, следует вычислить число мультивыражений, которые будут содержаться во всех несвязанных группах пространства поиска.

Следует отметить, что запрос любой топологии можно представить в виде полностью соединенного запроса, достроив недостающие связи путем добавления фиктивного предиката соединения, который всегда имеет значение «истина». Добавленные таким образом связи по определению будут представлять декартовы произведения соответствующих отношений. Все группы в пространстве поиска для полностью соединенного запроса являются связанными. Следовательно, для получения числа мультивыражений в несвязанных группах для запроса некоторой топологии T1 необходимо найти разность между числом мультивыражений в пространстве поиска для построенного на его основе полностью соединенного запроса и числом мультивыражений в пространстве поиска для исходного запроса топологии T1.

Для этого в работе выполнена оценка размеров пространства поиска для различных типов деревьев поиска и топологий исходного запроса. Число мультивыражений в несвязанных группах пространства поиска $N_{не\_связ}$ определяет, насколько меньше мультивыражений будет генерировать оптимизатор по сравнению с полностью соединенным запросом $N_{пс}$. Отношение $N_{не\_связ}/N_{пс}$ может трактоваться как *вероятность отсечения* группы $P_{отс}$ (исходя из классического определения вероятности).

Данная величина зависит от типа создаваемых деревьев поиска и топологии исходного запроса. Ниже приведены формулы для оценки вероятности отсечения группы для трех случаев.

1. Кустовое дерево поиска / линейный запрос:

$$P_{отс} = (N_{пс}–N_{лин})/N_{пс} = (3^n - 2^{n+1} - \frac{n^3 - n + 3}{3}) / (3^n - 2^{n+1} + n + 1). \quad (1)$$

2. Лево-глубокое дерево поиска / линейный запрос:

$$P_{отс} = (N_{пс}–N_{лин})/N_{пс} = (2^{n-1} - n) / 2^{n-1}. \quad (2)$$

3. Лево-глубокое дерево поиска / звездообразный запрос:

$$P_{отс} = (N_{пс}–N_{лин})/N_{пс} = ((n+1)2^{n-2} - 2n + 1) / n2^{n-1}. \quad (3)$$

На рис. 4 приведены вероятности отсечения $P_{отс}$ несвязанных групп для различных типов деревьев поиска и топологий исходного запроса, включающего от 3 до 13 соединяемых отношений. Из рисунка видно, что для линейных запросов при построении деревьев поиска любого типа вероятность отсечения приближается к 0.9, начиная с 7 соединяемых отношений. Для звездообразных запросов эта вероятность практически на зависит от числа соединяемых отношений и находится на уровне 0.5-0.55.

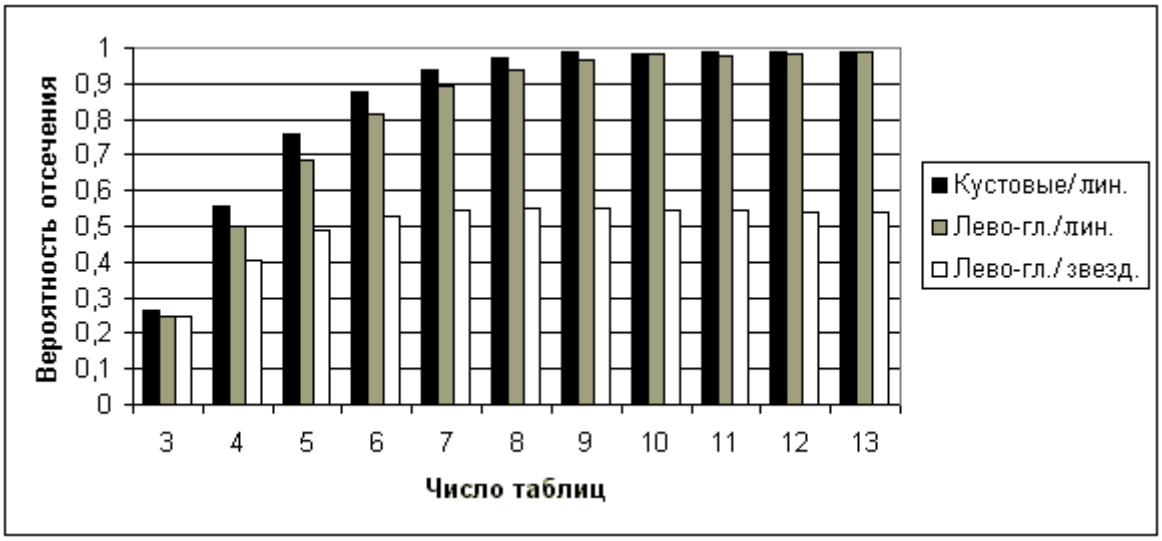

Рис. 4. Вероятности отсечения несвязанных групп в пространстве поиска.

Заключение

Разработаны два новых метода поиска оптимального плана на основе нисходящей стратегии анализа альтернативных планов. Первый позволяет увеличить значение стоимости мультивыражения ($C_{нг}$). Второй определяет метод генерации пространства поиска, при использовании которого для оптимизации связных запросов самый первый план выполнения в пространстве поиска не будет содержать декартовых произведений. Это позволяет уменьшить величину верхней границы стоимости группы ($C_{вг}$). Таким образом, разработанные методы поиска позволяют увеличить вероятность отсечения групп в процессе оптимизации.

Доказана лемма, определяющая достаточный набор правил для генерации полного пространства поиска для лево-глубоких исходных запросов. Доказана теорема, показывающая, что при использовании сформулирован-

ного набора правил для оптимизации связанных ациклических запросов все связанные группы в пространстве поиска будут начинаться с мультивыражений, не содержащих декартовых произведений.

Проведена количественная оценка эффективности разработанных методов поиска оптимального плана на основе нисходящей стратегии, показавшая, что их использование позволяет значительно увеличить вероятность отсечения несвязанных групп в пространстве поиска.

ЛИТЕРАТУРА

1. *Selinger P.G., Astrahan M.M., Chamberlin D.D., Lorie R.A., Price T.G.* Access Path Selection in a Relational Database Management System // Proc. ACM SIGMOD Int. Conf. Manag. Data. Boston, Mass., May 1979. P. 23-34.
2. *Lohman G.M., Guy M.* Grammar-Like Functional Rules for Representing Query Optimization Alternatives // Proceedings of the ACM SIGMOD Conference. Chicago, IL June 1988. P. 18-27.
3. *Graefe G., DeWitt D. J.* The EXODUS Optimizer Generator // Proc. SIGMOD, 1987. P. 160-172.
4. *Pirahesh H., Hellerstein J.M., Hasan W*. Extensible/Rule Based Query Rewrite Optimization in Starburst // Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data. San Diego, California. June 1992. P. 39-48.
5. *Carey M., Shekita E., Lapis G., Lindsay B., McPherson J.* An Incremental Join Attachment for Starburst // Proc. VLDB 1990. P. 662-673.
6. *Haas L., Kossman D., Wimmers E., Yang J*. Optimizing Queries Across Diverse Data Sources // Proc. VLDB, 1997. P. 276-285.
7. *Ono K., Lohman G.M*. Measuring the Complexity of Join Enumeration in Query Optimization // Proc. VLDB, 1990. P. 314-325.
8. *Gassner P., Lohman G.M., Schiefer K.B*. Query Optimization in IBM's DB2 Family of DBMSs, IEEE Data Engineering Bulletin, 16(4), 1993. P. 4-18.
9. *Vance B., Maier D*. Rapid Bushy Join-order Optimization with Cartesian Products // Proc. SIGMOD, 1996. P. 35-46.
10. *Graefe G., McKenna W.J*. The Volcano Optimizer Generator: Extensibility and Efficient Search // Proc. Data Engineering Conf., 1993. P. 209-218.
11. *Kabra N., DeWitt D*. OPT++: an object-oriented implementation for extensible database query optimization // VLDB Journal: Very Large Data Bases. 8(1). May 1999. P. 55-78.
12. *Graefe G.* The Cascades Framework for Query Optimization // Bulletin of the IEEE Technical Committee on Data Engineering. 18(3). 1995. P. 19-29.
13. *Graefe G*. The Microsoft Relational Engine // Proc. Data Engineering Conf., 1996. P. 160-161.
14. *Celis P*. The Query Optimizer in Tandem's ServerWare SQL Product // Proc. VLDB, 1996. P. 592.
15. *Pellenkoft M., Galindo-Legaria C., Kersten M*. The Complexity of Transformation-Based Join Enumeration // Proc. VLDB, 1997. P. 306-315.