Оценка эффективности минимизации ограничений запросов к СУБД

Кузнецов С. Д., kuzloc@ispras.ru, Мендкович Н. А., mend@f-group.ru

Аннотация: Эта статья описывает усовершенствованные алгоритмы лексической оптимизации запросов. Алгоритмы обнаруживают и удаляют избыточные условия из ограничения запроса, чтобы упростить его. Статья также представляет результаты применения этих оптимизационных техник и их влияние на скорость обработки запроса.

Ключевые слова: оптимизация запросов, лексическая оптимизация, выполнение запросов.

1. Введение

Вследствие эволюции вычислительной техники в течение последних десятилетий увеличивается роль таких приложений, как системы управления базами данных (СУБД). Важную роль в функционировании современных СУБД играют высокоуровневые языки запросов. Они позволяют формулировать и эффективно решать различные задачи, связанные с поиском и обработкой информации, хранящейся в базах данных.

С ростом объема хранимых данных и усложнением управленческих процессов, которые требуют выполнения сложных запросов, затрагивающих большие объемы хранимых данных, нетривиальной становится задача организации БД и повышения эффективности обработки запросов. Возникают новые проблемы функционирования СУБД, касающиеся недостаточной быстроты выполнения мощных операций, необходимых для обеспечения приемлемого уровня производительности системы. В особенности это касается повышения скорости обработки запросов на выборку данных. В рамках разрешения последней проблемы в области исследования баз данных выделилась область называемая *оптимизацией запросов*, которой посвящено значительное число работ [1-5].

Для дальнейшего изложения необходимо условиться о значении используемых терминов. Будем называть *запросом* любое языковое выражение, которое описывает совокупность данных в базе данных (БД), подлежащих выборке или обновлению. В запросе мы выделяем *ограничение* – часть запроса, содержащую все множество условий, описывающих выборку,

выдачей которой должно завершаться его исполнение. *Условие* – часть ограничения запроса, описываемая с помощью множества атрибутов запроса и констант и содержащая не более чем одну операцию сравнения. Каждое условие рассматривается как лексическая конструкция, поэтому в его рамках выделяется *корень* – множество входящих в его состав атрибутов. Помимо корня в состав условия входят операции преобразования и сравнения значений, а также константы.

Исследователи предлагают значительное число различных алгоритмов, в том числе, основанных на генерации и поиске оптимального плана доступа, выборе наиболее экономных потенциальных и низкоуровневых процедур и модификации самого текста запроса с целью приведения к наиболее оптимальной форме.

Последний вид оптимизации (модификация) подразделяется на два вида оптимизирующих преобразований: семантические, основанные на анализе содержимого БД и существующих в ней ограничений целостности, и лексические, оперирующие исключительно текстом самого запроса. Последний класс операций может быть признан наиболее широко применимым особенно в случае распределенной обработки запроса, так как оптимизация осуществляется без обращения данным БД и не связана с издержками при передаче данных.

Существует три вида лексической оптимизации запросов:

-перезапись (в частности, перемещение элементов по дереву запроса[6, Pp. 354-366], удаление вложенных подзапросов и проч. [7, Pp. 147-152; 8, Pp. 91-102]);

-украшение (введение в текст запроса дополнительных элементов ускоряющих обработку, т.н. «магические множества»[9]);

-сокращение (исключение избыточности в условии запроса).

Последнему классу оптимизирующих алгоритмов в современных работах уделяется недостаточно внимания[10, Сс. 204-207], несмотря на то, что не исчерпаны возможности повышения эффективности оптимизации и расширения множества поддающихся обнаружению и устранению видов избыточности.

Судя по опубликованным данным о принципах работы оптимизаторов широко распространенных СУБД, с целью удаления избыточных условий в них реализуются только алгоритмы поиска общих подвыражений в конъюнктах условий. В частности, это касается MySQL[11-12], PostgreSQL[13] и Oracle[14, P. 9].

Отказ от иных способов оптимизации, например, обработки дизъюнктов ограничений, можно рассмотреть на примере работы оптимизатора из описания MySQL 5.5[12], где при описании приемов лексической оптимизации приводится пример преобразования:

((a AND b) AND c OR (((a AND b) AND (c AND d)))) = >

(a AND b AND c) OR (a AND b AND c AND d).

Однако очевидно, что данное преобразование не завершено и итоговое кратчайшее представление данной функции

a AND b AND c,

так как X OR (X AND a) = > X[1].

Подобная логическая незавершенность преобразования ограничения демонстрирует наличие значимых нерешенных проблем в области идентификации избыточных условий в запросах.

В целях обнаружения, идентификации и устранения избыточности нами разработана процедура, включающая 3 основных операции:

- оптимизация на основе поглощения дублирующих друг друга условий в составе запроса;
- оптимизация набора условий с использованием алгоритмов минимизации булевой алгебры;
- оптимизация ограничения с помощью алгоритмов линейной алгебры.

Описание представленных ниже алгоритмов ранее публиковалось авторами[15-17]. В данной главе их описание приводится комплексно с представлением примеров реализации.

2. Описание алгоритмов

2.1. Алгоритм поглощения

Задача поглощения элементарных избыточных условий на основе поиска общих подвыражений в ограничении поставлена еще в наиболее ранних работах, посвященных оптимизации запросов[18, P. 247].

Эти преобразования являются одной из «ключевых технологий трансформации» запросов в оптимизаторах ряд СУБД, включая Oracle[14, P. 21; 19, P. 1368] и MySQL[11]. Однако они сводятся к достаточно тривиальному поиску избыточных условий, удаление одного из которых не изменяет смысла ограничения, но делает его более лаконичным.

Однако в существующих публикациях отсутствует подробное описание работы этих алгоритмов, поэтому неясно, сколь широкий круг избыточностей они могут устранять. Приводимые в упомянутых источниках примеры позволяют предположить, что указанные СУБД могут находить различные

[1]Данная ошибка присутствует во всех изданиях СУБД MySQL, включая версию для MySQL 5.5, изданную в 2013 году.

избыточные выражения среди условий, являющихся строгими тождествами (*arith-expr1 comp-op arith-expr2*). Наконец, в упомянутых источниках отсутствует подробное описание критериев выявления избыточности и процедуры их устранения, что затрудняет оценку эффективности этих алгоритмов.

В этом разделе описывается алгоритм, позволяющий находить и сокращать широкий набор условий, являющихся избыточными, но не совпадающих текстуально даже после стандартизации выражения. Для этого требуется перейти от уровня формального синтаксиса выражений до их семантики, сравнивая их части между собой. Избыточность может выражаться в том, что один из конъюнктов ограничения, записанного в ДНФ[2], избыточен по отношению к другому, так как все значения, описанные первым конъюнктом, входят в число значений, описанных вторым конъюнктом.

В рамках поиска таких избыточностей производится сравнение конъюнктов, содержащих условия с одинаковой левой частью (*arith-expr1*), и приводится их логическое сопоставление с целью выявления случаев, когда «однокоренные» условия могут поглощать друг друга из-за отсутствия условий, не являющихся общими или не поглощающими друг друга.

Например, запрос

```
SELECT ID_пользователя
FROM EMP
WHERE (ID>0 AND Salary>500 AND Salary+Bonus<1000) OR (Salary>400);
```

является неоптимальным, и может быть представлен в более лаконичной и оптимальной форме:

```
SELECT ID_пользователя
FROM Пользователи
WHERE Salary>400;
```

Для данного вида оптимизации сформулирована система правил поглощения одними условиями других в рамках описанной модели. Эти правила представлены в таблице 2.1, где первая строка соответствует поглощающему условию, первый столбец – поглощаемому, а в каждом из полей представлено необходимое и достаточное условие для указанного поглощения. Прочерк означает невозможность поглощения в данном случае.

[2]В рамках нашего алгоритма все условия запроса приводятся к дизъюнктивной нормальной форме, поэтому, обсуждая структуру ограничения, мы оперируем иерархией – условие, конъюнкт, ограничение в виде дизъюнкта конъюнктов.

Таблица 2.1. Правила поглощения условий.

	y == const2	y > const2	y < const2
x == const1	const2 == const1	const1 > const2	const2 > const1
x > const1	-	const1 > const2	-
x < const1	-	-	const2 < const1

Более подробное описание алгоритма поглощения представлено в другой нашей работе[16, Сс. 26-28].

2.2 Алгоритм Квайна

Ряд приемов по преобразованию логических выражений к более краткой форме разработан в рамках булевой алгебры. Они основаны на идее «склейки» пар конъюнктов, содержащих выражение и его отрицание при общих прочих одинаковых членах. Применимые для решения этой задачи способы минимизации логических функций были разработаны еще в начале 1950-х гг. [20-21], (т.е. еще до начала специальных исследований в области оптимизации запросов, которые относится к началу 1970-х гг.), позже они были адоптированы для программной реализации У. Квайном и И. Маккласки[22-23].

Возможность использования алгоритма Квайна-Маккласки при оптимизации запросов допускалась в ряде более поздних работ[24, P. 234; 25-26], однако описания примеров его реализации в данных целях нам обнаружить не удалось.

Для выполнения алгоритма Квайна требуется выделить в ограничении группу конъюнктов, содержащих одинаковое количество условий (например, n). Затем проводится попарное сравнение конъюнктов и «склейка» тех пар, которые имеют вид (A>B) AND X и NOT(A>B) NOT X, где X – некий конъюнкт условий размерностью n-1. Конъюнкты X именуются *первичными импликантами* и записываются отдельно. Затем для всех выбранных X повторяется процедура попарного анализа и «склейки» с целью получения конъюнктов размерностью n-2, и так до тех пор, пока удается обнаруживать пары конъюнктов вида (A>B) AND X и NOT(A>B) NOT X.

Мы предлагаем следующее программное представление данного алгоритма. Его целью является определение минимального множества импликант, соответствующего максимальному множеству конъюнктов, с целью их замены в конечном представлении. Для этого образуется таблица где строкам соответствуют первичные импликанты всех размерностей, а столбцам – конъюнкты исходного представления ограничения. Элемент таблицы имеет значение 0, если первичная импликанта данной строки не входит в конъюнкт, соответствующий столбцу, и 1 – если входит.

Для определения искомого набора импликант производятся пять этапов обработки этой таблицы:

1. Из дальнейшего анализа исключаются столбцы, содержащие только нули. Они в любом случае входят в конечное представление, поскольку не покрываются найденными импликантми.
2. Производится поиск *существенных импликант*, т.е. строк, одно из единичных значений которых является единственным в столбце. Обнаруженные существенные импликанты и соответствующие им конъюнкты исключаются из дальнейшего анализа
3. В видоизмененной таблице выделяются одинаковые столбцы, содержащие единицы в одних и тех же строках. Все столбцы-«дубликаты», кроме одного, исключаются из таблицы.
4. Исключаются все строки, не содержащие единичных значений (их образование возможно после завершения предыдущего этапа).
5. Из числа оставшихся строк выбирается набор импликант, включающий единицы во всех столбцах. Если имеется несколько вариантов, то выбирается тот, который содержит минимальное число условий.
6. Формируется итоговое представление ограничения, состоящие из конъюнктов, выбранных на этапе 1, и импликант, выбранных на этапах 2 и 5.

В рамках решения задачи оптимизации запросов данный алгоритм использует иное основание для «склейки» конъюнктов, нежели поиск пар «утверждение-отрицание». Учитывая, что в рамках анализа запросов происходят операции с непосредственным содержанием условий, в качестве критерия «склейки» мы используем признак тождественной истинности логической суммы двух утверждений.

Рассмотрим работу алгоритма на следующем примере. Пусть требуется отобрать авторов, имеющих различные соотношениям чисел опубликованных статей, книг и цитирований их работ другими авторами. Ограничение данного запроса будет иметь следующий вид:

...WHERE Articles>25 AND Books>2 AND Citations=32

OR NOT(Articles>25) AND Books>2 AND Citations=32

OR NOT(Articles>25) AND Books>2 AND NOT(Citations=32)

OR Articles>25 AND NOT(Books>2) AND NOT(Citations=32)

В результате применения описанного выше алгоритма выражение примет следующую более краткую форму:

Books>2 AND Citations=32

OR NOT(Articles>25) AND Books>2

OR Articles>25 AND NOT(Books>2) AND NOT(Citations=32)

2.3 Алгоритм оптимизации линейных неравенств

Также нами разработан алгоритм минимизации ограничений запроса содержащих условия с неодинаковой левой частью (*arith-expr1*), но имеющих общие атрибуты. Данный алгоритм аналогичен подходам, представленным в линейной алгебре, однако сам является абсолютно новым.

Большинство существующих в данной области работ посвящено методам минимизации систем линейных равенств[28]. К их числу относятся ставший классический алгоритм, известный как «метод Гаусса» или «метод Гаусса-Зейделя» (подробное описание представлено в [29, сс. 159-162]). Методы, разработанные для решения задач оптимизации систем линейных неравенств, кроме описанного ниже, – авторам неизвестны. Однако возникновение подобных задач применительно к оптимизации запросов представляется возможным.

Пусть имеется ограничение запроса, представленное в виде конъюнкции условий, которое направлено на то, чтобы отобрать значения, соответствующие некоему набору неравенств. Примером этого может служить следующее ограничение:

...WHERE (Salary+Bonus>10000

AND Salary-Payment>3000

AND 2*Payment>8000)

OR Payment-Bonus=>100,

оптимальной формой представления которого является выражение:

Payment-Bonus=>100

Алгоритм оптимизации подобного рода ограничений основан на попарном анализе всех условий конъюнкта, имеющих общие атрибуты. На основе этой процедуры создаются новые временные условия-«следствия». Кроме того, в текст включаются внешние условия, т.е. неравенства обратные одиночным условиям, объединенным с анализируемым конъюнктом операцией логического сложения (в данном примере это условие Payment-Bonus=>100).

Формулирование следствий осуществляется на основе системы формальных правил, часть из которых представлена в таблице 3. 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 следствие значимое для оптимизации.

Таблица 2.2. Общий вид определения следствий для пар неравенств, где левая часть парного полностью входит в рабочее неравенство.

	a+b= const1	(a-b=const1)	(a+b>const1)	(a+b<const1)	(a-b>const1)	(a-b<const1)
(b> const2)	a< const1- const2	a>const2+ const1	-	a<const1- const2	a>const1+ const2	-
(b< const2)	a> const1- const2	a<const2+ const1	a>const1- const2	-	-	a<const1+ const2
(a> const2)	b< const1- const2	b>const2- const1	-	b<const1- const2	-	b>const2- const1
(a< const2)	b> const1- const2	b<const2- const1	b>const1- const2	-	b<const2- const1	-

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

После серии преобразований анализируемый конъюнкт расширяется за счет включения условий следствий и внешних условий. После этого проводится формальной поиск логических противоречий и избыточных выражений, являющихся подмножеством следствий из других пар условий или внешних выражений, включенных в конъюнкт.

В итоге операции все следствия и внешние условия удаляются из конъюнкта вместе с выражениями, признанными избыточными. В случае обнаружения логических противоречий, делающих конъюнкт невыполнимым, из обработки исключается сам конъюнкт.

Более подробное описание реализации данного алгоритма представлено в предыдущей нашей работе[17, сс. 144-154].

3. Оценка эффективности алгоритма

С точки зрения практической эффективности преобразование запроса или плана его выполнения может привести к изменению стоимости его обработки в большую или меньшую сторону.

Возможны следующие определения эффективности алгоритма:

-оптимизирующий, т.е. ведущий к сокращению затрат на обработку запроса путем сокращения использования машинных ресурсов, превышающего стоимость выполнения алгоритма оптимизации;

-равноценный, описывающий случаи, когда экономия, связанная с оптимизацией запроса, и затраты на реализацию алгоритма – равны;

-убыточный, т.е. случай, когда расходы, связанные с выполнением оптимизирующего алгоритма, превышают достигнутую экономию, или же само осуществленное преобразование исходного запроса приводит к его «удорожанию» в сравнении с изначальной стоимостью.

С целью определения эффективности разработанных алгоритмов проведена серия экспериментальных испытаний их влияния на скорость обработки запросов, итоги которой приведены в настоящем разделе. Ее текст включает описание существующих критериев эффективности и обоснование выбора одного из них. Кроме того, дается описание тестовой среды, условий эксперимента и анализа его результатов.

3.1 Тестирование алгоритмов и оценки их эффективности

Для оценки эффективности разработанных алгоритмов необходимо проведение анализа стоимости, т.е. затрат системных ресурсов на их реализацию. Эффективность оптимизации определяется затратами на выполнение оптимизированного и неоптимизированного вариантов запроса с учетом затрат на осуществление самого процесса оптимизации.

В результате возникает задача определения ресурсов, использование которых должно быть минимизировано при выполнении запросов, и их затрат на реализацию плана доступа и алгоритма оптимизации. Традиционно в качестве меры для оценки эффективности алгоритмов используются следующие критерии [1, Pp. 113-114]:

1. затраты на текущее хранение данных (storage cost), связанные с использованием памяти;
2. стоимость передачи данных, т.е. обмен данными между машинным комплексом, в рамках которого осуществляется хранение, и местом вычисления (communication cost), а также время обмена данными между вторичной и основной памятью (secondary storage access cost или I/O cost);
3. затраты, характеризующие использование центрального процессора, в связи с вычислением запроса (computation cost).

Учитывая динамику развития вычислительных средств значительно сокращается дефицит ресурсов оперативной памяти, поэтому ключевыми становятся показатели, характеризующие временные затраты на выполнение алгоритма.

Была предложена следующая функция вычисления стоимости обработки запроса, где в качестве единицы измерения используется время[30, P. 233]:

$$T_{total}=T_{CPU}+T_{I/O}+T_{MSG}+T_{TR}, \qquad (3.1),$$

где T_{total} – общие временные затраты, T_{CPU} – процессорное время, $T_{I/O}$ – время обмена данными между вторичной и основной памятью, T_{TR} – время коммуникации, T_{MSG} – время, затраченное на инициирование запросов и сообщений, а также обработку данных, полученных от внешних ресурсов. В случае, когда речь идет о локальной базе данных, последние два компонента могут быть признанными равными 0.

Со временем роль каждой из составляющих стоимости обработки запросов была пересмотрена. Прежде всего, в связи с ростом объемов памяти вычислительных систем стоимость хранения данных перестала быть «узким местом» [1, P. 144] и играть значимую роль в оценке эффективности алгоритмов оптимизации.

Стоимость передачи данных является все еще важным фактором в функционировании информационных систем, так как все большую популярность приобретают распределенные базы данных, а также распределенные вычислительные системы, что значительно повышает роль издержек, связанных с обменом данными в процессе реализации запроса. В некоторых работах данный вид затрат рассматриваются как единственный критерий эффективности алгоритма (например, [31, P. 97]).

Однако в случае предложенных нами алгоритмов ситуация является иной. Все процедуры, связанные с оптимизирующими преобразованиями запроса, осуществляются локально и не используют какие-либо данные, хранящиеся в базе данных, к которой будет адресован запрос, или характеризующие ее состояние. Предложенные алгоритмы оптимизации могут быть названы «автономными». Естественно, они в силу своих свойств не могут привести к увеличению затрат, связанных с обменом данными, но могут привести к их сокращению в результате устранения из запроса избыточных условий, обработка которых потребовала бы увеличения коммуникационных издержек.

На основе выше изложенных аргументов в качестве критерия оценки эффективности работа алгоритмов оптимизации нами выбраны вычислительные затраты. Следующим шагом является выбор конкретных характеристик, отражающих работу ЦП по обработке запроса и реализации алгоритма оптимизации.

В современных исследованиях большой популярностью пользуется подход, ориентированный на учет числа машинных операций, с помощью которого характеризуются затраты процессорного времени. В современных исследованиях существуют два основных подхода к оценке стоимости с их помощь [32, P. 177]:

-унифицированная ценовая модель (uniform cost model) в качестве единицы измерения рассматривающая одну машинную операцию;

-и логарифмическая ценовая модель (logarithmic cost model), где стоимость каждой машинной операции корректируется с учетом объема вовлеченных данных.

Первый подход пользуется широкой популярностью в виду простоты и иногда считается фактически безальтернативным [33, Pp. 26-27]. Однако доказано, что он может приводить к значительному искажению оценки стоимости процесса из-за различий в объемах данных, необходимых при реализации алгоритма[32, Pp. 177-179].

В рамках решения данной проблемы в качестве критерия стоимости работы алгоритма в нашем исследовании используется **время выполнения задачи процессором**. Использование времени в качестве средства измерения стоимости в данном случае представляется оптимальным и с точки зрения решения проблем «унифицированной» модели, и как универсальный критерий, позволяющий охарактеризовать все виды затрат, связанных с обработкой запросов [30, Pp. 233-234].

3.2 Описание тестовой среды

В рамках экспериментальной проверки была проведена серия испытаний в специально созданной тестовой среде, включающей в себя СУБД Oracle Database 10g Express Edition, а также программный комплекс «Оптимизатор», оптимизирующий условия запросов к реляционным базам данных.

Указанные программные продукты были установлены на персональном компьютере, оснащенном процессором Intel (R) Core i5-2430M с частотой 2,4 Ггц, установленной памятью 4 Гбайт, 64 разрядной операционной системой Windows 7.

Выбор Oracle Database 10g определялся широким распространением данного приложения, встроенными приложениями измерения затрат системного времени, доступ пользователя к генерируемым планам доступа, что облегчало проведение экспериментов. Погрешность измерения времени средствами СУБД составляет 0,01 секунды.

Программный комплекс включает в себя приложения, производящие минимизацию числа условий запроса с помощью алгоритмов, описанных выше: сокращение запроса с помощью поглощения избыточных условий, оптимизация по Квайну-Маккласки, а также оптимизация запроса как системы неравенств. «Оптимизатор» включал в себя приложения, позволяющие оптимизировать ограничение запроса как по одному из перечисленных выше критериев, так и с помощью всех описанных алгоритмов. В итоге программа генерирует оптимальный вид ограничения запроса.

При отладке «Оптимизатора» была проведена серия тестов, в рамках которых приложение преобразовало серию произвольных выражений. Результаты этих испытаний мы приводим ниже в табл. 3.1. Время, затраченное системой на оптимизацию выражения, указывается в микросекундах (μs). Кроме того, в таблице приводятся данные о числе элементарных условий в изначальной форме оптимизируемого выражения.

Таблица 3.1. Результаты тестирования программы «Оптимизатор»

Тест	Время оптимизации (μs)	Число условий
1	122,913	4
2	91,0689	9
3	33,7007	24
4	52,5365	4
5	93,5856	9
6	81,6848	7
7	81,3892	7
8	52,4639	4
9	48,4827	4

На основе представленных данных можно заключить, что программа-оптимизатор совершает обработку при сравнительно небольших затратах времени.

Причем указанные затраты демонстрируют слабую зависимость от числа условий в оптимизируемом выражении. Линейный коэффициент корреляции, рассчитанный на основе данных табл. 3.1, равен 0,37, что указывает на отсутствие явно выраженной статистической зависимости. Различия во временных затратах на преобразование запросов обусловлены функционирование фоновых процессов в период работы программы. Однако создаваемая ими погрешность не превышает 100 μs, что не оказывает влияния на итоги тестов, описанных ниже (раздел 3.3). Кроме того, данная погрешность равна неустранимой погрешности оценки времени обработки запроса средствами СУБД Oracle Database 10g.

Перед проведением тестирования в рамках СУБД была создана группа таблиц данных, имеющих идентичный набор атрибутов и объем данных. Каждая из созданных таблиц включала атрибуты, представленные в таблице 3.2.

Таблица 3.2. Структура таблицы, используемой в тестовой среде.

Атрибут	Тип данных
Name	Char(100)
Date	Date
N1	Number
N2	Number
N3	Number
N4	Number

В каждой таблице размещено 100 записей, содержащих различные значения перечисленных атрибутов. С целью оценки производительности системы с учетом оптимизации запросов была проведена серия тестов, включающих реализацию запросов на выборку данных, ограничение которых содержало бы о 2 до 4 атрибутов таблицы.

Осуществление запросов и учет времени их реализации производится средствами самой СУБД.

3.3 Результаты тестирования

В данном разделе мы описываем проведение тестирования и его результаты. Оно осуществляется с помощью реализации ряда запросов к описанной выше базе данных в среде Oracle Database 10g.

Все запросы были направлены на выборку равного числа атрибутов из таблиц одинаковой размерности и отличались исключительно составом ограничения. Каждый из них содержал те или иные виды избыточности. В запросах не используются вложенные подзапросы. Ограничение состояло из множества условий, представленных в дизъюнктивной нормальной форме.

Порядок эксперимента имел следующий вид:

- реализация запроса в первичной форме (время выполнения T_1);
- выполнение программы оптимизации запроса (T_{opt});
- реализация оптимизированного запроса (T_2).

После проведения перечисленных испытаний проводилось сравнение системного времени, затраченного на реализацию каждого из этапов (в терминах формулы 3.1 - T_{total}).

Эффективность алгоритма определялась соотношением

$$I_{eff} = (T_2 + T_{opt})/T_1, \qquad (3.2)$$

где I_{eff} – коэффициент эффективности оптимизации.

Кроме того, целью эксперимента является оценка взаимосвязи числа условий в составе запроса со скоростью его обработки, для чего также проводится учет двух других показателей:

-число условий в составе запроса до оптимизации (A_1);

-число условий в составе запроса после оптимизации (A_2).

Результаты тестов представлены в таблице 3.4.

Таблица 3.3. Результаты тестов (время указывается в секундах).

	T_1	T_{opt}	T_2	I_{eff}	A_1	A_2
1	0,3	0,00008	0,1	0,3336	3	1
2	0,5	0,00011	0,1	0,2002	6	2
3	0,8	0,00008	0,15	0,1876	5	2
4	0,4	0,00010	0,2	0,5003	7	4
5	0,8	0,00008	0,2	0,2501	7	3
6	0,14	0,00007	0,03	0,2148	5	3
7	0,06	0,00008	0,01	0,1680	4	3
8	0,02	0,00009	0,009*	0,4545	9	6
9	0,23	0,00008	0,02	0,0873	3	1
10	0,15	0,00008	0,08	0,5339	3	1

*Менее 0,01 секунды, приводится максимально возможное значение из-за погрешности измерения.

Результаты тестирования показывают, что время обработки оптимизированного запроса значительно ниже, чем его обработка без проведения лексической оптимизации. Среднее ускорение обработки запроса времени в ходе проведенных тестов составляет 4,5 раза. (См. рисунок 1).

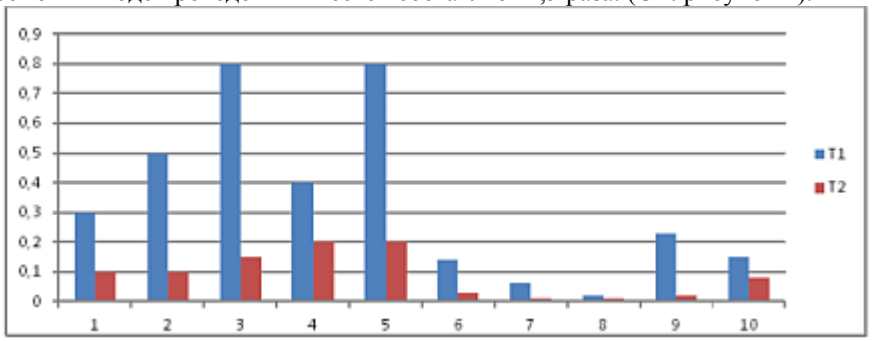

Рисунок 1. Время выполнения запроса до (T_1) и после оптимизации (T_2). (Время в секундах).

Коэффициент эффективности оптимизации демонстрирует возможность сокращения стомости обработки запроса на 47%-84%. Причем временные затраты на собственно оптимизацию запроса пренебрежимо малы в сравнении с затратами на его обработку. Проведенные тесты показывают, что они в 1000-10000 раз меньше, чем временные затраты системы на выполнение запроса.

Анализ причин роста скорости обработки запросов предполагает зависимость временных затрат от числа условий в составе ограничения запроса. Сравнение отношений показателей A_1 и A_2 с I_{eff} не позволяет установить однозначую статистическую зависимость, что, в частности, может быть обусловлено качественными отличиями между условиями и временем необходимым для их обработки.

Кроме того, статистическая зависимость может искажаться ошибкой измерения. При исключении из выборки тестов время обработки которых меньше 0,03 секунд (напомним, что точности измерения времени СУБД до 0,01 секунды), линейный коэффициент корреляции составляет 0,42, что демонстрирует прямую зависимость между этими двумя показателями.

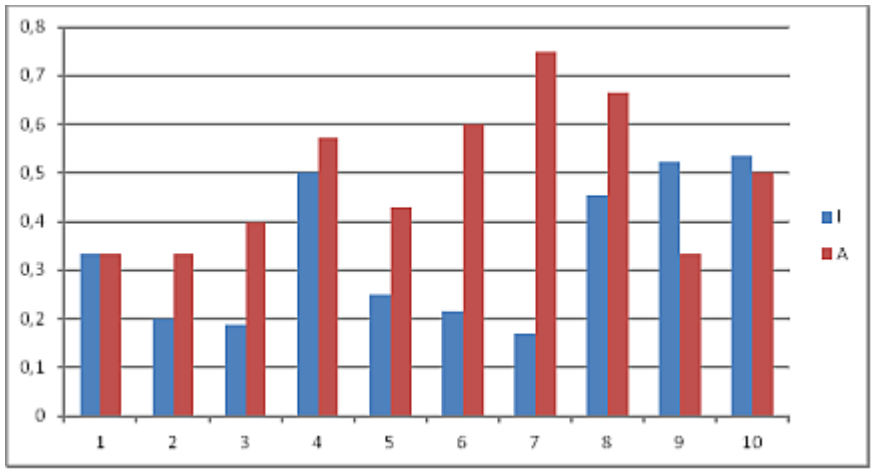

Рисунок 2. Коэффициент эффективности оптимизации запроса (I) и соотношение числа условий после и до проведения оптимизации ограничения (A).

Таким образом, проведенные тесты позволяют констатировать, что лексическая оптимизация запросов с помощью описанных выше алгоритмов позволяет сократить затраты на обработку запроса за счет сокращения избыточных условий.

При этом временные затраты на реализацию оптимизационных алгоритмов пренебрежимо малы в сравнении с временем обработки запроса СУБД. Это позволяет утверждать, что описанные в нашей работе средства лексической оптимизации позволяют повысить эффективность существующие систем оптимизации запросов.

Заключение

В данной работе были описаны алгоритмы лексической оптимизации запросов, позволяющие устранить виды избыточности, неопределимые существующими программами оптимизаторами.

Приводится подробное описание тестирования данных алгоритмов оптимизации и их влияния на стоимость обработки запроса. Анализируются существующие критерии оценки эффективности работа алгоритмов, осуществляется выбор критерия для данного исследования. Также описывается тестовая среда, включающая в себя СУБД Oracle Database 10g. В тестировании были учтены погрешности измерения и обработки, связанные с особенностями тестовой среды.

Проведенные тесты показывают, что временные затраты на реализацию разработанных алгоритмов лексической оптимизации запросов не превышают 0,0002 секунды, что в 1000 раз меньше времени выполнения рассмотренных запросов к базе данных. Это указывает на то, что выполнение указанных оптимизационных алгоритмов даже в случаях, когда оно не ведет к оптимизации запроса, не может вызвать значимое замедление его обработки.

Список литературы

[1]. Jarke M., Koch J. Query Optimization in Database Systems // ACM Computing Surveys (CSUR), 1984. March, Volume 16, Issue 2. Pp. 111-152.

[2]. Mannino M. V., Chu P., Sager T. Statistical profile estimation in database systems // ACM Computing Surveys, Volume 20, Issue 3. September 1988.

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

[4]. Chaudhari S. An Overview of Query Optimization in Relational Systems // Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 1998.

[5]. Кузнецов С. Д. Методы оптимизации выполнения запросов в реляционных СУБД. [http://www.citforum.ru/database/articles/art_26.shtml]. [Обращение 20 ноября 2012].

[6]. Chaudhuri S., Shim K. Including Group-By in Query Optimization // Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann, San Mateo, USA, 1994. San Francisco: Morgan Kaufmann Publishers Inc., 1994. Pp. 354-366.

[7]. 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. Jeju Island: IEEE, 2009. Pp. 147-152.

[8]. 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. San Francisco: Morgan Kaufmann Publishers Inc., 1992. Pp. 91-102.

[9]. Sippu S., Soisalon-Soininen E. An Analysis of Magic Sets and Related Optimization Strategies for Logic Queries // Journal of the ACM, Volume 43, № 6, November 1996.

[10]. Мендкович Н. А., Кузнецов С. Д. Обзор развития методов лексической оптимизации запросов // Труды Института системного программирования, т. 23, М., ИСП РАН, 2012. Сс. 204-207.
[11]. 7.2.1.2 How MySQL Optimizes WHERE Clauses [http://dev.mysql.com/doc/refman/5.5/en/where-optimizations.html]. [Обращение 20 ноября 2012].
[12]. MySQL 5.5 Reference Manual. Chapter 7. Optimization. http://dev.mysql.com/doc/refman/5.5/en/optimization.html, [Опубликовано в 2010 году, обращение 20 ноября 2012].
[13]. Пакет PostgreSQL 8.3.3. Адрес файла postgresql-8.3.3\src\backend\optimizer\util\predtest.c.
[14]. Query Optimization in Oracle Database 10g Release 2. An Oracle White Paper, June 2005. Redwood Shores, Oracle Corporation, 2005.
[15]. Mendkovich N., Kuznetcov S. New Algorithms for Lexical Query Optimization // Proceedings of the 31st International Conference on Information Technology Interfaces. Cavtat/Dubrovnik, Croatia, June 22-25, 2009. Zagreb: University of Zagreb, 2009. Pp. 187-192.
[16]. Кузнецов С. Д., Мендкович Н. А. Новые алгоритмы лексической оптимизации запросов // Модели и анализ информационных систем, 2009. Т. 16, № 4. С. 22-33.
[17]. Мендкович Н. А., Кузнецов С. Д. Оптимизация конъюнктов условий в составе запросов // Модели и анализ информационных систем, 2011. Т. 18, № 3. С. 144-154.
[18]. Hall P. A. V. Optimization of single expressions in a relational data base system // IBM Journal of Research and Development, 1976. Volume 20, Number 3.
[19]. Bellamkonda S., Ahmed R., Witkowski A., Amor A., Zait M., Lin Ch.-Ch. Enhanced Subquery Optimizations in Oracle / // Proceedings of the 35th international conference on Very large data base, August 2009. Volume 2 Issue 2. P. 1368.
[20]. Veitch E. W. A Chart Method for Simplifying Truth Functions // ACM Annual Conference/Annual Meeting: Proceedings of the 1952 ACM Annual Meeting, Pittsburg: ACM, NY, 1952.
[21]. Karnaugh M. The Map Method for Synthesis of Combinational Logic Circuits // Transactions of the American Institute of Electrical Engineers. Part I, № 72 (9), November 1953.
[22]. Quin W. V. O cores and prime implicants of truth functions // American Mathematics Monthly. 1959. V. 66. №9.
[23]. McCluskey E. J. Minimization of Boolean Functions // The Bell System Technical Journal. November 1956. V. 35, Issue 5.
[24]. Wu M. C. Query Optimization for Selecting Using Bitmaps // ACM SIGMOD Record Volume 28 Issue 2, June 1999. P. 234.
[25]. Тарасенко П. Ф., Бухарова М. Ф. Технология «The Reporter» для построения отчетов по базам данных // Вестник Томского Государственного Университета, № 275, апрель 2002.
[26]. Sarma A., Theobald M., Widom J. Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. Technical Report. Stanford, 2007. [http://ilpubs.stanford.edu:8090/800/] [Обращение 20 ноября 2012].
[27]. Saad Y., van der Vorst H. A. Iterative solution of linear systems in the 20th Century // Journal of Computational and Applied Mathmatics, Issue 123, 2000.

[28]. Benzi M. The Early History of Matrix Iterations: With a Focus on the Italian Contribution // SIAM Conference on Applied Linear Algebra, Monterey Bay, Seaside, California, 26 October 2009.
[29]. Бронштейн И. Н., Семедяев К. А. Справочник по математике для инженеров и учащихся втузов. 13-е изд. исправленное. М.: Наука, 1986.
[30]. Ozsu M. N., Valduriez P. Principles of Distributed Database Systems. Second Edition. New Jersey, 1999. P. 233.
[31]. Graefe G. Query Evaluation Techniques for Large Databases // ACM Computing Surveys, 1993. Volume 25, Issue 2.
[32]. Hromkovi J. Theoretical Computer Science: Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2004.
[33]. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. Third Edition. Cambridge-London: Massachusetts Institute of Technology Press, 2009.

Minimization of data base query's conditions: evoluation of efficiency

Kuznetsov S. D., kuzloc@ispras.ru, Mendkovich N. A., mend@f-group.ru

Abstract: This paper describes enhanced algorisms of query lexical optimization. These algorisms detect and remove redundant conditions from query restriction to simplify it. The paper also presents results of implementation of these optimization techniques and those effects on query processing speed.

Key words: query optimization, lexical optimization, query processing.

Сведения об авторах:

Сергей Дмитриевич Кузнецов,
Институт Системного Программирования РАН,
профессор, главный научный сотрудник;

Никита Андреевич Мендкович,
ООО Объединение сетей ФРИнет, инженер.