УДК 004.65
https://doi.org/10.36906/KSP-2022/58

Мамедли Р.Э.
канд. физ.-мат. наук
Нижневартовский государственный университет
Нижневартовск, Россия

СТАТИЧЕСКАЯ ОПТИМИЗАЦИЯ РАСПРЕДЕЛЕННЫХ ЗАПРОСОВ В ШИРОКОВЕЩАТЕЛЬНЫХ СЕТЯХ

Аннотация. Оптимизация запросов является одним из наиболее актуальных проблем реализации территориально распределенных баз данных (БД). Представленная работа посвящена изучению вопроса, связанного с алгоритмом оптимизации запросов в распределенных БД. Автором применяются эмпирические и теоретические методы исследования. В работе используются научные материалы отечественного и зарубежного авторства.

Ключевые слова: распределенные базы данных; статическая оптимизация запросов; распределенная обработка запросов; широковещательные сети; SQL.

Mammadli R.E.
Ph.D.
Nizhnevartovsk State University
Nizhnevartovsk, Russia

STATIC OPTIMIZATION OF DISTRIBUTED QUERIES ON BROADCAST NETWORKS

Abstract. Query optimization is one of the most relevant problem in the implementation of geographically distributed databases. The presented work is devoted to the study of the issue related to the query optimization algorithm in distributed databases. The author uses empirical and theoretical research methods. The work uses scientific materials of domestic and foreign authorship.

Keywords: distributed databases; static query optimization; distributed query processing; broadcast networks; SQL.

Распределенная база данных (БД) – это несколько логически взаимосвязанных баз данных, которые расположены в узлах распределенной вычислительной системы [7]. Структура распределенной БД определяет оптимальную стратегию размещения фрагментов БД для обеспечения целостности, безопасности и производительности. Стратегия распределения определяет, как разделить БД и где хранить каждый фрагмент [2]. Узлы распределенной системы могут быть разных конфигураций. Связи между ними тоже могут

быть различными. Доступ к данным и управление ими требует особого внимания со стороны распределенной системы управления базами данных (СУБД) [1; 3-6].

Когда отношения или их части хранятся в разных узлах, для получения ответов на запросы, требуется выполнять операции соединения или объединения.

В территориально распределенных системах характерны большие задержки при передаче сообщений и более высокая частота ошибок. Современные распределенные системы управления базами данных скрывают низкоуровневые детали физической организации данных. Но когда производительность становится критически важной, обработка запросов привлекает много внимания. В распределенных системах оптимизация запросов намного труднее из-за большого количества факторов, влияющих на производительность. Запросы к фрагментированным отношениям, реплицированным базам данных увеличивают затраты на передачу данных. Количество узлов тоже влияет на время получения ответа.

При оптимизации запросов перед пользователями стоят две основные цели: уменьшение времени передачи данных и уменьшение времени ответа. Большинство алгоритмов оптимизации игнорируют затраты на передачу данных целевому узлу. В данной статье будем принимать во внимание фрагментацию. Для наглядности рассмотрим только вертикальную фрагментацию.

Рассмотрим распределенную базу данных Нижневартовского государственного университета, где данные студентов находятся в разных департаментах (узлах). Предположим, что данные содержатся в отношениях СТУДЕНТ, ДИСЦИПЛИНА и ЭКЗАМЕН и фрагментированы как показано ниже:

Узел 1	Узел 2	Узел 3
СТУДЕНТ	ДИСЦИПЛИНА	ЭКЗАМЕН

Необходимо получить список студентов, допущенных к экзамену по дисциплине СУБД. Для нераспределенных систем на языке SQL данный запрос выглядел бы как в Листинге 1:

```
SELECT СТУДЕНТ.СТФИО
FROM СТУДЕНТ
INNER JOIN ЭКЗАМЕН ON СТУДЕНТ.СТКОД = ЭКЗАМЕН.СТКОД
INNER JOIN ДИСЦИПЛИНА ON ДИСЦИПЛИНА.ДСКОД = ЭКЗАМЕН.ДСКОД
WHERE ДИСЦИПЛИНА.ДСНАЗВ=`СУБД`
```

Листинг 1. Локальный SQL-запрос

Для соединения трех отношений возможны четыре сайта-кандидата: сайт первого отношения, сайт второго отношения, сайт третьего отношения или четвертый сайт. Для межсайтовой передачи данных поддерживаются два метода:

1. Отправка целиком. Все отношения отправляются на сайт присоединения и сохраняются во временном отношении до присоединения. Если алгоритм соединения представляет собой соединение слиянием, отношения не нужно сохранять, и сайт соединения может обрабатывать входящие кортежи в конвейерном режиме по мере их поступления.

2. Извлечение по мере необходимости. Внешние отношения последовательно сканируются, и для каждого кортежа значение соединения отправляется на сайт внутреннего отношения, который выбирает внутренние кортежи, совпадающие со значением, и отправляет выбранные кортежи на сайт внешнего отношения. Этот метод эквивалентен полусоединению внутреннего отношения с каждым внешним кортежем.

В распределенной системе для выполнения данного запроса необходимо учитывать размер данных в отношениях и выбрать одну из следующих стратегий:

1. переместить ДИСЦИПЛИНА и ЭКЗАМЕН на узел 1 и выполнить запрос (СТУДЕНТ ⋈ ЭКЗАМЕН ⋈ ДИСЦИПЛИНА);

2. переместить СТУДЕНТ и ЭКЗАМЕН на узел 2 и выполнить запрос (СТУДЕНТ ⋈ ЭКЗАМЕН ⋈ ДИСЦИПЛИНА);

3. переместить СТУДЕНТ и ДИСЦИПЛИНА на узел 3 и выполнить запрос (СТУДЕНТ ⋈ ЭКЗАМЕН ⋈ ДИСЦИПЛИНА);

4. переместить СТУДЕНТ, ДИСЦИПЛИНА и ЭКЗАМЕН на узел 4 и выполнить запрос (СТУДЕНТ ⋈ ЭКЗАМЕН ⋈ ДИСЦИПЛИНА);

Алгоритм оптимизации прогнозирует общее время каждой стратегии и выбирает самую минимальную. Учитывая, что после объединения СТУДЕНТ ⋈ ЭКЗАМЕН ⋈ ДИСЦИПЛИНА не выполняется никаких операций, очевидно, что стратегия 4 сопряжена с наибольшими затратами, поскольку должны быть переданы все отношения. Если размер (СТУДЕНТ) намного больше, чем размер (ДИСЦИПЛИНА) и (ЭКЗАМЕН), стратегии 2 и 3 минимизируют время связи и, вероятно, будут лучшими, если локальное время обработки не слишком велико по сравнению со стратегиями 1 и 4.

Если стратегия 2 не является лучшей, выбор делается между стратегиями 1 и 3. Затраты на локальную обработку в обеих этих альтернативах идентичны. Если отношение СТУДЕНТ имеет большой размер и только несколько кортежей в отношениях ДИСЦИПЛИНА и ЭКЗАМЕН совпадают по размеру, стратегия 1, вероятно, требует наименьшего времени на обмен данными и является наилучшей. В противном случае, то есть, если отношение СТУДЕНТ небольшого размера или совпало много кортежей в отношения ДИСЦИПЛИНА и ЭКЗАМЕН, необходимо выбрать между стратегиями 2 и 3.

Для решения данной задачи необходимо решить проблему оптимизации, правильно выбрать отношения для перемещения и узлы, в которых будет осуществляться обработка. Для запроса с *n*-отношениями *n–1* отношений предстоит переместить в узел, где находится оставшееся отношение, и реплицировать. Для параллельного выполнения можно разбить отношения на *k* фрагментов. На выбор нужных отношений и узлов влияет еще и топология сети. В широковещательных сетях репликация выполняется легче, чем в сетях с двухточечным

соединением. Количество промежуточных узлов тоже необходимо учитывать, так как при увеличении количества узлов уменьшается время ответа из-за параллельной обработки, но возрастает полное время из-за увеличения затрат на передачу данных.

Таким образом, для минимизации времени передачи данных и времени обработки необходимо учитывать местоположение отношений, их размеры и тип сети.

Рассмотрим правила минимизации времени передачи данных. Допустим, в сети M узлов и отношений. R_iобозначает отношении R, хранящийся в узле i. Время передачи b байтов на k (1≤k≤m) узлов обозначена $T(b)$. На широковещательных сетях:

$$T_k(b) = T_1(b)$$

Тогда правило можно сформулировать в виде, как в Листинге 2:

ЕСЛИ $size(R_i) > max_{j=1,n}\left(size\left(R_j\right)\right)$ **ТОГДА**

обрабатывающим будет узел i с наибольшим объемом данных

ИНАЧЕ

R_p наибольшее отношение **and** узел R_p является обрабатывающим

Листинг 2. Правило определение узла-процессора

Концептуально алгоритм можно рассматривать как исчерпывающий поиск среди всех альтернатив, которые определяются перестановкой порядка соединения отношения, методов соединения (включая выбор алгоритма соединения), результирующего узла, пути доступа к внутреннему отношению и межузлового взаимодействия. режим передачи. Такой алгоритм имеет комбинаторную сложность по количеству задействованных отношений. На самом деле алгоритм значительно сокращает количество альтернатив за счет использования динамического программирования и эвристики. При динамическом программировании дерево альтернатив строится динамически и сокращается путем исключения неэффективных вариантов.

Оценка производительности алгоритма в контексте как высокоскоростных сетей (аналогичных локальным сетям), так и среднескоростных глобальных сетей подтверждает значительный вклад затрат на локальную обработку даже для глобальных сетей. В частности, показано, что для распределенного соединения передача всех внутренних отношений лучше, чем метод выборки по мере необходимости.

В данной статье рассмотрен статический подход оптимизации запросов в распределенных СУБД. Сначала сформулирована задача распределенной обработки запросов. Основное предположение состоит в том, что входной запрос выражен на языке реляционной алгебры. Сложность задачи пропорциональна выразительной способности языка запросов и его способности к абстрагированию. Проблема решается с помощью статического подхода, когда оптимизатор запросов взаимодействует с исполняющей средой до этапа выполнения.

Литература

1. Грофф Дж., Вайнберг П., Оппель Э. SQL: Полное руководство. М.: Вильямс, 2015. 961 с.

2. Мамедли Р.Э. Системы управления базами данных. Нижневартовск: Изд-во Нижневарт. гос. ун-та, 2021. 214 с.

3. Мейер М. Теория реляционных баз данных. М.: Мир, 1987. 608 с.

4. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. Cengage Learning, Inc, 2019. 837 p.

5. Fagin R. The Decomposition Versus Synthetic Approach to Relational Database Design // Proceedings of the 3rd International Conference on Very Large Data Bases. Vol. 3. VLDB Endowment, 1977. Pp. 441–446. (VLDB ’77).

6. Kossmann D., Stocker K. Iterative Dynamic Programming: A New Class of Query Optimization Algorithms // ACM Transactions on Database Systems. 2000. Vol. 25. № 1. Pp. 43–82. https://doi.org/10.1145/352958.352982

7. Tamer Özsu M., Valduriez P. Principles of Distributed Database Systems. Springer Science+Business Media, LLC, 2011.