УДК 004.9
DOI: 10.26102/2310-6018/2022.36.1.020

Алгоритмизация повторной оптимизации запросов в облачных базах данных на основе компьютерного обучения

О.А.Р. Аль Мусави[1], О.Я. Кравец[2]✉

[1]Васитский университет, Эль Кут, Ирак
[2]Воронежский государственный технический университет, Воронеж, Российская Федерация
csit@bk.ru✉

Резюме: В облачных средах конфигурация оборудования, использование данных, и распределение рабочей нагрузки постоянно меняются. Эти изменения затрудняют оптимизатору запросов системы управления облачными базами данных подобрать оптимальный план выполнения запроса (QEP). Чтобы оптимизировать запрос с более точной оценкой затрат, в литературе было предложено во время выполнения запроса осуществлять повторную оптимизацию запроса. Тем не менее, некоторые из этих оптимизаций не могут обеспечить прирост производительности с точки зрения времени ответа на запрос или денежных затрат, которые являются двумя целями оптимизации для облачных баз данных, и могут оказывать негативное влияние на производительность из-за накладных расходов. Это поднимает вопрос о том, как определить, когда оптимизация выгодна. Целью исследования является разработка метода повторной оптимизации запросов, который использует компьютерное обучение. Ключевая идея алгоритма заключается в использовании прошлых выполнений запросов, чтобы научиться прогнозировать эффективность повторной оптимизации запросов, и делается это с целью помочь оптимизатору запросов избежать ненужной повторной оптимизации запросов для будущих запросов. Метод осуществляет запрос поэтапно, используя модель компьютерного обучения, для прогнозирования того, будет ли повторная оптимизация запроса полезной после выполнения этапа, и вызывает оптимизатор запросов для автоматического выполнения повторной оптимизации. Предстоит экспериментальная оценка эффективности.

Ключевые слова: повторная оптимизация запросов, облачные базы данных, компьютерное обучение, многоэтапный запрос, автоматизация исполнения

Для цитирования: Аль Мусави О.А.Р., Кравец О.Я. Алгоритмизация повторной оптимизации запросов в облачных базах данных на основе компьютерного обучения. *Моделирование, оптимизация и информационные технологии.* 2022;10(1). Доступно по: https://moitvivt.ru/ru/journal/pdf?id=1147 DOI: 10.26102/2310-6018/2022.36.1.020

Algorithmization of repeated query optimization in cloud databases with the aid of computer training

O.A.R. Al Musavi[1], O.Ya. Kravets[2]✉

[1]University of Wasit, El-Cut, Iraq
[2]Voronezh State Technical University, Voronezh, Russian Federation
csit@bk.ru✉

Abstract: In cloud environments, hardware configuration, data usage, and workload distribution are constantly changing. These changes make it difficult for the query optimizer of the cloud database management system to choose the optimal query execution plan (QEP). In scientific literature, it was proposed to re-optimize the query during its execution for the purpose of optimizing it with a more accurate cost estimate. However, some of these optimizations cannot provide performance gains in terms

of query response time or monetary costs, which are the two optimization goals for cloud databases, and may have a negative impact on performance due to overhead. This raises the question of how to determine when the optimization is efficient. The aim of the study is to develop a method of repeated query optimization that uses computer training. The key idea of the algorithm is to employ past query executions to learn how to predict the effectiveness of query re-optimization, and this is done in order to help the query optimizer avoid unnecessary re-optimization of queries for future ones. The method runs the query step-by-step, utilizing a computer training model, to predict whether re-optimization of the query will be useful after the stage is completed, and calls the query optimizer to automatically perform re-optimization. An experimental evaluation of the effectiveness is to be carried out.

Keywords: repeated query optimization, cloud databases, computer training, multi-stage query, automation of execution

For citation: Al Musavi O.A.R., Kravets O.Ya. Algorithmization of repeated query optimization in cloud databases with the aid of computer training. *Modeling, Optimization and Information Technology*. 2022;10(1). Available from: https://moitvivt.ru/ru/journal/pdf?id=1147 DOI: 10.26102/2310-6018/2022.36.1.020 (In Russ).

Введение

Одно из ключевых различий между оптимизацией запросов в традиционных базах данных и в облачных базах данных заключается в том, что оптимизация запросов в облачных базах данных направлена на снижение денежных затрат, выплачиваемых провайдеру облачных услуг в дополнение к времени ответа на запрос. Тем не менее, временные и денежные затраты, необходимые для выполнения запроса, оцениваются на основе статистических данных, доступных оптимизатору запросов в момент выполнения оптимизации запроса. Эти статистические данные часто приблизительны, что может привести к неточным оценкам временных и денежных затрат, необходимых для выполнения запроса [1, 2]. Таким образом, планы выполнения запросов (QEP), сгенерированные оптимизатором запросов на основе этой статистики, до выполнения запроса, могут быть не самыми лучшими.

1. Материалы и методы

Одним из подходов, который может быть применен для решения вышеуказанной проблемы, является адаптивная обработка запросов [3]. Эта стратегия состоит не в выполнении запроса в целом за один раз, а вместо этого разделяет выполнение каждого запроса на несколько этапов, и затем повторно запускает оптимизатор запросов между каждым этапом. Сделав это, оптимизатор запросов может собирать более точную статистику между этапами выполнения, которые могут позволить изменять QEP во время выполнения, таким образом, вероятно, повышая производительность запросов [1, 4]. Операторы, которые не полагаются на выполнение других, группируются вместе, и такие группы называются “Этапами”. Например, если план запроса содержит оператор JOIN, его левая и правая части выполняются на отдельном этапе. После выполнения каждого из этапов QEP, статистика данных обновляется, чтобы оптимизатор запросов мог использовать самую последнюю статистику для создания улучшенного (т.е. повторно оптимизированных) QEP для тех этапов, которые еще предстоит выполнить. В результате повторной оптимизации запроса, этапы QEP, которые еще не были выполнены, могут измениться, поскольку операторы в этих QEP могли бы быть заменены другими или поскольку любой этап может быть перепланирован для запуска на другой машине. Такие изменения в QEP могут привести к различному времени ответа на запрос и различным денежным затратам. Однако многократные вызовы оптимизатора запросов во время выполнения запроса имеют связанные накладные расходы, которые,

в свою очередь, приводят к дополнительным денежным затратам. По этой причине желательно повторно оптимизировать запрос только в том случае, если повышение стоимости повторно оптимизированного QEP поверх оригинала QEP может возместить затраты, возникшие при многократном вызове оптимизатора.

На любом приведенном этапе выполнения запроса решение о том, приведет ли повторная оптимизация к повышению производительности, является непростой задачей. В [3] такое решение принимается на основе эвристических методов. Несколько контрольных точек размещаются вручную между определенными типами операторов. Тогда разница между предполагаемыми затратами и фактическими затратами считается после проверки контрольной точки запроса. Если эта разница превышает заранее определенный порог, то происходит повторная оптимизация. Проблема такой методики заключается в том, что правила размещения контрольных точек и порог фиксированы. Из-за динамики облачных сред сроки повторной оптимизации, определенные с помощью данного метода, недостаточно точны, чтобы сократить время выполнения запроса. В [5] представлен алгоритм обработки запросов, который выполняет повторную оптимизацию запросов после завершения каждого этапа на основе методики, предложенной в [1]. Однако работа [5] показывает, что многие из этих вызовов повторной оптимизации не привели к изменению базового QEP, что означает, что повторная оптимизация запроса была выполнена без необходимости. Это произошло из-за того, что этапы не были согласованы с наилучшим временем для применения повторной оптимизации. Например, после выполнения примера запроса 1, приведенного на Рисунке 1, установлено, что из 10 раз, когда оптимизатор вызывался для повторной оптимизации во время выполнения этого запроса, только 2 из этих вызовов изменили QEP для оставшихся этапов; поэтому большинство вызовов повторной оптимизации не привели ни к улучшению времени, ни денежных затрат на выполнение этого запроса.

```
SELECT R.p_id, R.p_name, R.sc, S.p_hr
FROM (SELECT p_id, p_name, AVG(p_bp) AS sc
FROM patient GROUP BY p_id, p_name) AS R
JOIN (SELECT p_id, p_hr
FROM patient
WHERE UDF(p_id,p_hr) > 80
) AS S
ON R.p_id = S.p_i
```

Рисунок 1 - Запрос 1
Figure 1 - Query 1

Естественно, вызов процедуры повторной оптимизации без необходимости увеличивает как время ответа на запрос, так и денежные затраты. Поэтому проблема заключается в определении наиболее подходящего времени для запроса на повторную оптимизацию и в определении тех случаев, когда повторная оптимизация может негативно повлиять на производительность запросов. Для решения этой проблемы в статье представлен основанный на компьютерном обучении алгоритм для повторной оптимизации запросов в облаке. Ключевая идея этого алгоритма заключается в использовании прошлых выполнений запросов, чтобы научиться прогнозировать эффективность повторной оптимизации запросов, и делается это с целью помочь оптимизатору запросов избежать ненужной повторной оптимизации запросов для будущих запросов.

2. Использование компьютерного обучения при обработке запросов

В то время как компьютерное обучение использовалось для улучшения обработки запросов в [6], известно, оно не использовалось для предотвращения ненужных вызовов повторной оптимизации запросов при адаптивной обработке запросов. Среди проблем, которые необходимо решить при использовании компьютерного обучения для этой цели, следующие. Первая состоит из множества функций, влияющих на оценку стоимости запроса, таких как избирательность, мощность, минимальные и максимальные значения столбца, наиболее частое значение столбца, гистограмма и т.д. Сложность здесь заключается в выборе наиболее подходящего подмножества из всех этих функций. Вторая проблема заключается в большом количестве возможных моделей компьютерного обучения. Алгоритмы контролируемого обучения, такие как Случайный лес [7], Нейронная сеть [8] и Расчет опорных векторов [9] широко используются, но для поставленной цели их необходимо тщательно изучить. Третья проблема касается сбора исторических данных по выбранному подмножеству функций, необходимых для обучения модели прогнозирования, построенной с использованием выбранного алгоритма компьютерного обучения. Четвертая проблема заключается в измерении эффективности алгоритма обучения. Работы [10, 11, 12] показывают, что алгоритм обучения эффективен для их собственных целей, например, для улучшения оценки затрат, но на самом деле ни один из них не демонстрирует, что они эффективны в фактической производительности выполнения запросов.

3. Повторная оптимизация запросов – описание и эксперименты

В [5], после отправки запроса в СУБД, обычный оптимизатор запросов сначала генерирует начальный QEP. Затем этот QEP будет разделен на этапы и поэтапно выполнен механизмом выполнения. После завершения этапа статистика данных обновляется. Эти статистические данные включают мощность, избирательность, а также максимальные и минимальные значения для каждого атрибута в каждой таблице базы данных. Обновляя эти статистические данные, оценка результирующего объема данных, используемого на последующих этапах, обновляется соответствующим образом. Остальные этапы QEP также отправляются оптимизатору запросов для повторной оптимизации с использованием обновленной статистики. Кроме того, в этой системе несколько машин с различными конфигурациями оборудования используются параллельно для выполнения операторов запросов. В настоящей статье эти машины будут называться контейнерами. Выполнение запроса на различные контейнеры приводит к различному времени ответа на запрос и денежным затратам, а при выборе наилучшего QEP необходимо учитывать и то, и другое. Для этого используем нормализованную взвешенную аддитивную модель [13] для выбора наилучшего плана. В этой модели каждая возможная альтернатива QEP оценивается по шкале, которая сочетает в себе как цели, время ответа на запрос, так и денежные затраты, с весами, определенными пользователем и средой для каждой цели, и определенным пользователем допустимым максимальным значением для каждой цели. Следующая функция $A_i^{WSM-score}$ используется для вычисления оценки QEP:

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j \frac{a_{ij}}{m_j}$$

где a_{ij} – значение альтернативы QEP с индексом i (QEP_i) для цели j, m_j – заданное пользователем допустимое максимальное значение для цели j, а w_j-нормализованный суммарный вес пользователя и среды для цели j, который определяется следующим

образом:

$$w_j = \frac{u_{w_j} e_{w_j}}{\sum_{j=1}^{n} (u_{w_j} e_{w_j})}$$

Здесь u_{w_j} и e_{w_j} описывают вес пользователя и вес окружающей среды для цели j, соответственно. Эти веса определяются пользователем. Поскольку различные цели отражают различные затраты, модель выбирает альтернативу с наименьшим ядром, чтобы минимизировать затраты.

Проведены эксперименты, сравнивающие производительность запросов, полученную в результате использования повторной оптимизации запросов, и без повторной оптимизации запросов. Использовано два набора машин. Первый набор состоит из одной локальной машины (двухъядерный процессор Intel i5 2500K, 3 ГГц и 16 ГБ DRAM), используемой для обучения модели машинного обучения и выполнения оптимизации запросов. Второй набор состоит из 10 выделенных виртуальных частных серверов (VPS), которые использовались для развертывания механизма выполнения запросов. Пять из этих VPS - небольшие контейнеры (процессор Intel Xeon E5 2682, 2,5 ГГц с 1 ГБ DRAM). Остальные 5 VPS – большие контейнеры (двухъядерный процессор Intel Xeon E5-2682, 2,5 ГГц с 2 ГБ DRAM). Оптимизатор запросов и механизм запросов, использованные в этом эксперименте, были модифицированы из PostgreSQL 8.4. Данные были распределены между этими VPS.

В экспериментах было создано 1200 запросов с использованием шаблонов запросов, представленных в [1]. Запрос 1, показанный на Рисунке 1, является запросом, созданным на основе одного из этих шаблонов. Результаты показывают, что использование повторной оптимизации в среднем на 20% улучшает общие временные затраты по сравнению с использованием без повторной оптимизации, в то время как денежные затраты на два подхода близки, с разницей всего в 4%. Эти увеличения денежных затрат обусловлены тем фактом, что более мощные контейнеры, выбранные для выполнения запросов, являются контейнерами, за которые поставщики облачных услуг взимают почасовую плату за выполнение запроса.

Однако большое количество повторной оптимизации запросов не требуется. Ненужная повторная оптимизация для QEP происходит, когда QEP не изменяется после выполнения повторной оптимизации. В этих экспериментах после выполнения этапа QEP автоматически выполняется повторная оптимизация независимо от того, изменилась ли статистика данных после выполнения этапа, что приводит ко многим ненужным повторным оптимизациям. Например, эксперименты показывают, что при выполнении первого запроса 8 из 10 повторных оптимизаций запроса не требуются. Только две необходимые повторные оптимизации происходят после выполнения подзапроса. За исключением этих случаев, QEP после выполнения оператора TableScan или Aggregate вообще не меняются после повторной оптимизации. Выполнение повторной оптимизации сопряжено с накладными расходами, а ненужная повторная оптимизация увеличивает время ответа на запросы и денежные затраты. В этих экспериментах почти 60% повторной оптимизации запросов являются ненужными, а выполнение повторной оптимизации одного запроса стоит около 0,5% от общего времени ответа на запрос.

Если запрос повторно оптимизируется только тогда, когда изменения в его QEP после повторной оптимизации могут быть гарантированы, то ненужной повторной оптимизации не будет. Чтобы обнаружить такие изменения, в следующем разделе описан метод повторной оптимизации на основе компьютерного обучения, предсказывающий, изменится ли QEP после повторной оптимизации на основе

исторических (предыдущих) данных выполнения запроса, и проводится повторная оптимизация для QEP только тогда, когда такое изменение прогнозируется.

4. Предлагаемая повторная оптимизация запросов на основе компьютерного обучения

4.1. Особенности, лежащие в основе алгоритма

На Рисунке 2 показаны основные этапы обработки запросов при включении алгоритма для повторной оптимизации запросов. Сначала оптимизатор получает запрос и записывает текущую статистику данных. Затем запрос компилируется в QEP с информацией об этапе. Первый этап в QEP выполняется и удаляется из QEP. Во время выполнения статистика данных отслеживается и обновляется. После выполнения первого этапа эти обновленные статистические данные сравниваются с текущими статистическими данных, которые были записаны до выполнения этапа. Модель компьютерного обучения используется здесь для учета разницы между текущей статистикой данных и новой статистикой данных в качестве входных данных и принятия решения о повторной оптимизации (“ДА” или “НЕТ”) в качестве выходных данных. Запрос повторно оптимизируется, если решение “ДА” и выполняется текущий первый этап в новом QEP после повторной оптимизации; в противном случае, если решение “НЕТ”, QEP остается прежним и выполняется его следующий этап. Эта процедура продолжается до тех пор, пока не останется ни одного этапа.

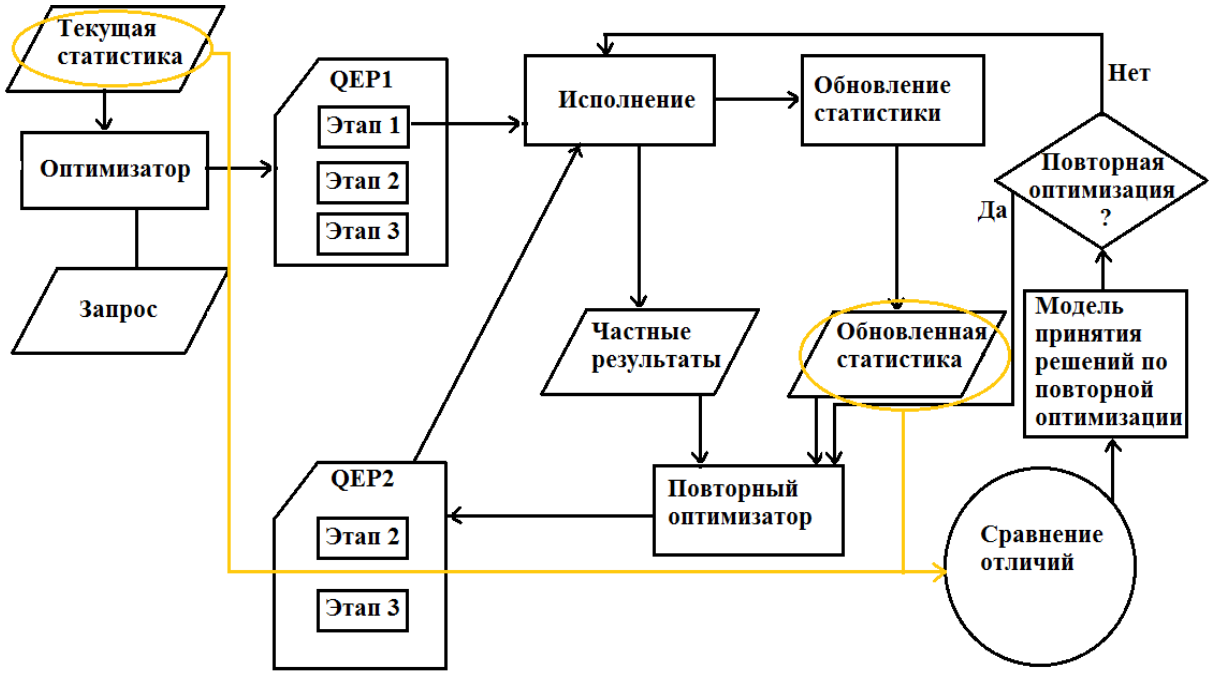

Рисунок 2 - Обработка запросов с повторной оптимизацией запросов на основе компьютерного обучения

Figure 2 - Query processing with repeated query optimization based on computer training

4.2. Выбор функций

В этом разделе определены статистические данные, необходимые для обучения модели компьютерного обучения.

Изменение QEP после повторной оптимизации подразумевает, что повторная оптимизация выгодна. Определим типы такого изменения:

1) изменения в типах физических операторов,
2) изменения в количестве контейнеров,
3) изменения в типах контейнеров.

Это означает, что если произойдет какой-либо из этих трех типов изменений, то следует разрешить повторную оптимизацию.

Предположим, что в текущей СУБД существуют столбцы C1, C2, ..., Cn во всех таблицах. Различия в избирательности (DIFF_SELECTIVITY), в количестве различных значений (DIFF_NDV) и в гистограммах (DIFF_HISTOGRAM) каждого столбца до и после выполнения этапа используются в качестве объектов данных в обучающих данных, используемых для прогнозирования, как показано в Таблице 1. Двоичное значение ДА/НЕТ используется в качестве прогнозируемого класса в обучающих данных, где ДА означает, что повторная оптимизация, по прогнозам, будет полезной, и никак иначе. Ранее установлено, что избирательность, количество различных значений и гистограмма влияют на оценку размера данных [11]. Таким образом, различия в этих трех функциях до и после выполнения этапа приводят к изменениям в оценке размера данных промежуточных результатов. Следовательно, они становятся актуальными при принятии решения об эффективности повторной оптимизации.

Таблица 1 - Список выбранных функций
Table 1 - List of selected functions

DIFF_SELECTIVITY(C1)	DIFF_NDV(C_1)	DIFF_HISTOGRAM(C_1)
DIFF_SELECTIVITY(C2)	DIFF_NDV(C_2)	DIFF_HISTOGRAM(C_2)
DIFF_SELECTIVITY(Cn)	DIFF_NDV(C_n)	DIFF_HISTOGRAM(C_n)

4.3. Модельное обучение

Прежде всего генерируются запросы для обучения модели, выполняя случайные запросы, сгенерированные из всех 22 типов запросов в тесте TPC-H [14], которые являются значениями функций, выбранных в разделе 4.2. Таким образом, модель прогнозирования может быть применена ко всем запросам. Если повторная оптимизация предназначена только для самых дорогостоящих/наиболее представительных запросов, то на этом первом этапе обучающие данные должны быть собраны из выполнения только случайных, но наиболее дорогостоящих/представительных запросов. На Рисунке 3 показана процедура сбора обучающих данных. Для того, чтобы лучше объяснить подробно как собираются обучающие данные, продемонстрирован пример выполнения запроса 2 (Рисунок 4).

После отправки запроса записывается текущая статистика Statcurr, собранная из системных журналов. Затем запрос отправляется оптимизатору для создания QEP. Этот QEP включает в себя информацию об этапах и узлах, на которых будут выполняться эти этапы. На Рисунке 5 показан QEP, созданный оптимизатором запросов для запроса 2. Каждый узел обозначает отдельный оператор запроса. Стрелки указывают на поток данных между операторами. QEP разделен на этапы, каждый из которых обозначен прямоугольником.

.

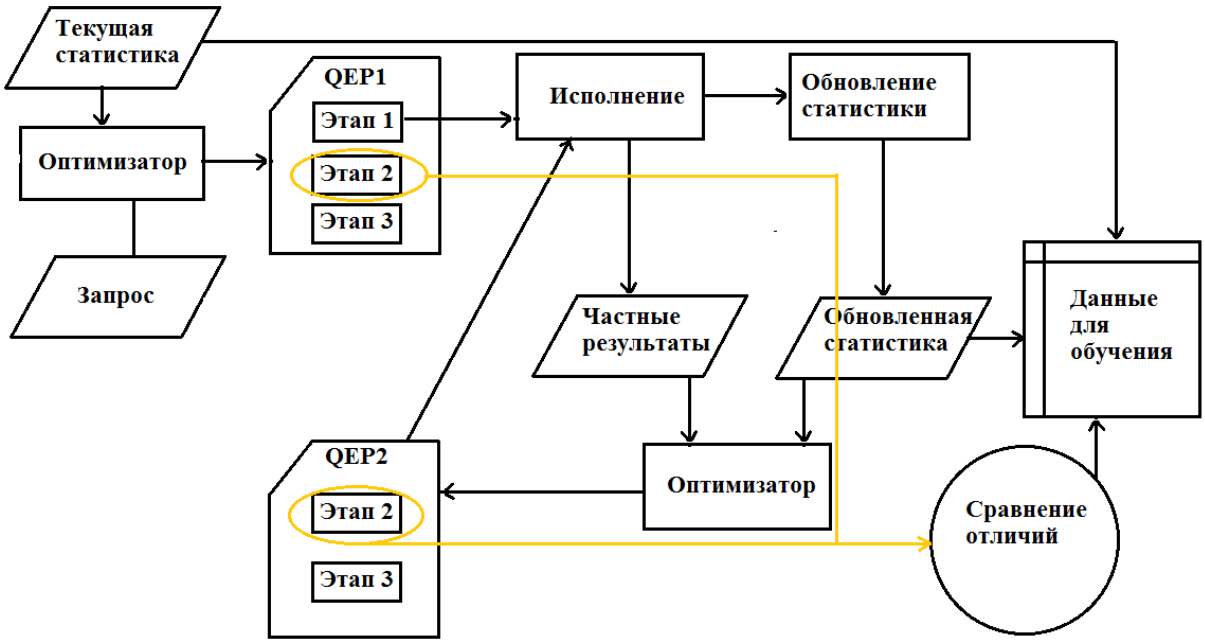

Рисунок 3 - Порядок сбора данных для обучения
Figure 3 - The order of data collection for training

```
SELECT Department, COUNT(Name)
FROM STUDENT
GROUP BY Department
WHERE Grade <= 'C';
```

Рисунок 4 - Запрос 2
Figure 4 - Query 2

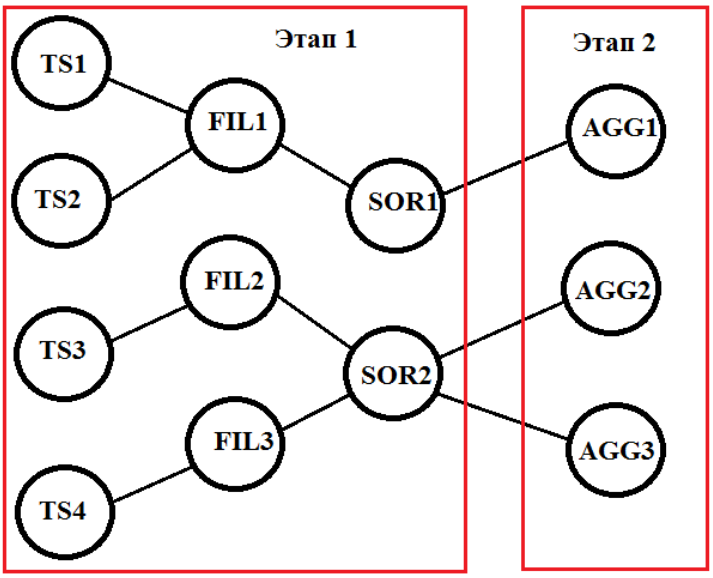

Рисунок 5 - QEP, разделенный на различные этапы, генерируемые оптимизатором запросов для запроса 2
Figure 5 - QEP divided into various stages which are generated by the query optimizer for Query 2

TS, FIL, SOR и AGG, обозначают операторы сканирования таблиц, сортировки, фильтрации и агрегирования соответственно

Результаты

В облачной системе баз данных, поскольку данные распределены между различными контейнерами, индексы различают одни и те же операторы, которые выполняются параллельно для разных данных в разных контейнерах. Затем этап 1 отправляется в механизм выполнения запросов. Во время выполнения обновляется статистика данных, используя метод, представленный в [1]. В этом методе статистика данных собирается во время выполнения и обновляется после завершения работы операторов в одной вершине. Обновленные статистические данные - Statupdate. Поскольку эти статистические данные собираются из фактически выполняемого запроса, Statupdate является более точным, чем Statcurr, который получается из оценки. Разница между Statupdate и Statcurr есть Statdiff. Statdiff включает значения функций, используемых в качестве обучающих данных. Например, текущая избирательность и обновленная избирательность столбца A равны 0,5 и 0,1 соответственно, затем разница 0,4 добавляется в качестве значения функции DIFF_SELECTIVITY в обучающем наборе данных. Этот процесс применяется ко всем функциям Таблицы 1.

Обсуждение

Если прогнозируется, что повторная оптимизация будет полезной, QEP затем повторно оптимизируется с использованием обновленной статистики данных. После этого выполняется следующий этап (Этап 2) на основе нового QEP. Затем процесс повторяется для остальных этапов. В этом примере возможно изменение этапа 2. На этом этапе после повторной оптимизации результат сравнивается с этапом 2 до оптимизации, чтобы увидеть любые потенциальные изменения.

Заключение

Целью исследования является разработка метода повторной оптимизации запросов, который использует компьютерное обучение. Ключевая идея алгоритма заключается в использовании прошлых выполнений запросов, чтобы научиться прогнозировать эффективность повторной оптимизации запросов, и делается это с целью помочь оптимизатору запросов избежать ненужной повторной оптимизации запросов для будущих запросов. Метод осуществляет запрос поэтапно, используя модель компьютерного обучения, для прогнозирования того, будет ли повторная оптимизация запроса полезной после выполнения этапа, и вызывает оптимизатор запросов для автоматического выполнения повторной оптимизации. Предстоит экспериментальная оценка эффективности.

СПИСОК ИСТОЧНИКОВ

1. Bruno N., Jain S., Zhou J. Continuous cloud-scale query optimization and processing. *VLDB Endow*. 2013;11(6):961–972.
2. Wolf F., May N., Willems R., Sattler K.-U. On the Calculation of Optimality Ranges for Relational Query Execution Plans. *2018 International Conference on Management of Data (SIGMOD '18)*. 2018;663–675.
3. Deshpande A., Ives Z., Raman V. Adaptive Query Processing. *Foundations and Trends in Databases*. 2017;1(1):1–140.
4. Markl V., Raman V., Simmen D., Lohman G., Pirahesh H., Cilimdzic M. Robust query processing through progressive optimization. *ACM SIGMOD International Conference on Management of data (SIGMOD '04)*. 2004;659–670.
5. Details omitted for double-blind reviewing.

6. Bankole A.A., Ajila S.A. Predicting cloud resource provisioning using machine learning techniques. *26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2013;1–4.
7. Breiman L. Random Forests. *Machine Learning.* 2001;45(5):5–32.
8. Schmidhuber J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2014;61(10):85–117.
9. Corinna C., Varnik V.N. Support-vector networks. *Machine Learning*. 1995;20:273–297.
10. Liu H., Xu M., Yu Z., Corvinelli V., Zuzarte C. Cardinality Estimation Using Neural Networks. *25th Annual International Conference on Computer Science and Software Engineering (CASCON '15)*. 2015;53–59.
11. Kipf A., Kipf T., Radke B., Leis V., Boncz P., Kemper A. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *VLDB Endow*. 2019;15(1):85–97.
12. Saravanan T., Shohedul H., Nick K., Gautam D. Approximate Query Processing for Data Exploration using Deep Generative Models. *36th International Conference on Data Engineering (ICDE)*. 2020;1309–1320.
13. Helff F., Gruenwald L., d'Orazio L. Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments. *EDBT/ICDE Workshops*. 2016;1558:751–760.
14. Barata M., Bernardino J., Furtado P. An Overview of Decision Support Benchmarks: TPCDS, TPC-H and SSB. *Advances in Intelligent Systems and Computing.* 2015;353:619–628.

REFERENCES

1. Bruno N., Jain S., Zhou J. Continuous cloud-scale query optimization and processing. *VLDB Endow*. 2013;11(6):961–972.
2. Wolf F., May N., Willems R., Sattler K.-U. On the Calculation of Optimality Ranges for Relational Query Execution Plans. *2018 International Conference on Management of Data (SIGMOD '18)*. 2018;663–675.
3. Deshpande A., Ives Z., Raman V. Adaptive Query Processing. *Foundations and Trends in Databases*. 2017;1(1):1–140.
4. Markl V., Raman V., Simmen D., Lohman G., Pirahesh H., Cilimdzic M. Robust query processing through progressive optimization. *ACM SIGMOD International Conference on Management of data (SIGMOD '04)*. 2004;659–670.
5. Details omitted for double-blind reviewing.
6. Bankole A.A., Ajila S.A. Predicting cloud resource provisioning using machine learning techniques. *26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2013;1–4.
7. Breiman L. Random Forests. *Machine Learning.* 2001;45(5):5–32.
8. Schmidhuber J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2014;61(10):85–117.
9. Corinna C., Varnik V.N. Support-vector networks. *Machine Learning*. 1995;20:273–297.
10. Liu H., Xu M., Yu Z., Corvinelli V., Zuzarte C. Cardinality Estimation Using Neural Networks. *25th Annual International Conference on Computer Science and Software Engineering (CASCON '15)*. 2015;53–59.
11. Kipf A., Kipf T., Radke B., Leis V., Boncz P., Kemper A. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *VLDB Endow*. 2019;15(1):85–97.
12. Saravanan T., Shohedul H., Nick K., Gautam D. Approximate Query Processing for Data Exploration using Deep Generative Models. *36th International Conference on Data Engineering (ICDE)*. 2020;1309–1320.

13. Helff F., Gruenwald L., d'Orazio L. Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments. *EDBT/ICDE Workshops*. 2016;1558:751–760.
14. Barata M., Bernardino J., Furtado P. An Overview of Decision Support Benchmarks: TPCDS, TPC-H and SSB. *Advances in Intelligent Systems and Computing.* 2015;353:619–628.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Аль Мусави Осама Адил Рахим, преподаватель**,** Васитский университет, Эль-Кут, Ирак
e-mail: oalmusawi@uowasit.edu.iq

Almusawi Osamah Adil Raheem, Teacher, University Of Wasit, El-Cut, Muhfaza Wasit, Iraq

Кравец Олег Яковлевич, д.т.н., Воронежский государственный технический университет, Воронеж, Российская Федерация.
e-mail: csit@bk.ru
ORCID: 0000-0003-0420-6877

Kravets Oleg Yakovlevich, Doctor Of Technical Sciences, Voronezh State Technical University, Voronezh, Russian Federation

Статья поступила в редакцию 13.02.2022; одобрена после рецензирования 24.02.2022; принята к публикации 09.03.2022.

The article was submitted 13.02.2022; approved after reviewing 24.02.2022; accepted for publication 09.03.2022.