МЕТОД ВЫБОРА ПЛАНА ВЫПОЛНЕНИЯ SQL ЗАПРОСА К РЕЛЯЦИОННОЙ СУБД

METHOD FOR SELECTING THE EXECUTION PLAN FOR AN SQL QUERY TO A RELATIONAL DATABASE MANAGEMENT SYSTEM

УДК-004

Ермаков Максим Сергеевич, бакалавр МГТУ им. Н. Э. Баумана, Россия, г. Москва

Yermakov Maxim Sergeevich, ermakov.1828@gmail.com

Аннотация

В данной статье описывается метод, в котором задача выбора плана выполнения SQL запроса сводится к задаче машинного обучения и тем самым решается проблема оценки количества кортежей, отбираемых условиями запроса. Данный метод позволит более точно предсказывать количество кортежей, следовательно, планировщик будет выбирать более подходящие к условиям в запросе, планы выполнения SQL запроса. Целью работы является разработка и программная реализация метода выбора плана выполнения SQL запроса к реляционной СУБД. В результате был разработан метод выбора плана запроса в реляционной СУБД, основанный на методе k-ближайших соседей и на учете статистики выполнения предыдущих запросов.

Annotation

This article describes a method in which the task of selecting an execution plan for an SQL query is reduced to a machine learning task, and thus solves the problem of estimating the number of tuples selected by query conditions. This method allows you to more accurately predict the number of tuples, so the scheduler will choose more appropriate SQL query execution plans for the conditions in the

query. The purpose of this work is to develop and implement a method for selecting the execution plan for SQL queries to a relational database management system. As a result, we developed a method for selecting a query plan in a relational DBMS based on the k-nearest neighbor method and taking into account the statistics of previous queries.

Ключевые слова: базы данных; машинное обучение; планировщик запросов; выбор плана запроса.

Keywords: databases; machine learning; query scheduler; query plan selection.

Определения и обозначения

План выполнения SQL запроса – конкретный ряд операций, которые необходимо выполнить для получения результата запроса. Такой план можно представить в виде дерева, ребра дерева представляют из себя движение данных, узлы – операции над данными.

Планировщик SQL запросов в реляционной СУБД – часть СУБД, которая отвечает за создание итогового плана выполнения запроса. В подавляющем большинстве современных реляционных СУБД используется стоимостной планировщик запросов.

Стоимостной планировщик – планировщик, который каждому возможному плану выполнения запроса определяет в соответствие некий числовой параметр - стоимость, по которому в дальнейшем из множества всех возможных планов выбирает план с минимальным значением стоимости.

Стоимость плана – объем ресурсов, необходимых для выполнения плана, то есть для получения всех кортежей результата.

Стоимость выполнения вершины дерева плана – стоимость выполнения поддерева этой вершины плана.

Мощность вершины – количество кортежей, возвращаемых вершиной плана. Важно отметить, что вершина возможно будет содержать условия, выполнение которых будет проверено для всех кортежей, возвращаемых этой вершиной.

Выборочность вершины – отношение мощности плана к количеству всех кортежей проходивших проверку условий этой вершиной.

Постановка проблемы

На сегодняшний день, в таких распространенных в использовании СУБД как PostgreSql, MySql, Oracle MSSQL, используется планировщик запросов, основанный на стоимостной модели. Ее суть заключается в том, что для каждого плана возможно оценить стоимость его выполнения и на основании этого критерия из всего множества планов СУБД выберет тот, который будет исполняться быстрее всего. В стоимостной модели каждая вершина дерева плана может выполнять 5 операций:

Таблица 1.

Коэффициенты доступа к диску

Краткое обозначение	коэффициент	Значение
c_s	seq_page_cost	1.0
c_r	random_page_cost	4.0
c_t	cpu_tuple_cost	0.01
c_i	cpu_index_tuple_cost	0.005
c_o	cpu_operator_cost	0.0025

Процесс выбора плана запроса состоит из этапа поиска плана во множестве всех возможных планов выполнения запроса и этапа оценки стоимости каждого плана. Однако, этап поиска не является узким местом в данном процессе, а сильное влияние на производительность оказывает именно этап оценки стоимости плана.

Процесс оценки стоимости в свою очередь, так же состоит из двух подзадач, которые решаются последовательно, из-за чего задача оценки стоимости решается настолько плохо, насколько плохо решается любая из этих двух подзадач.

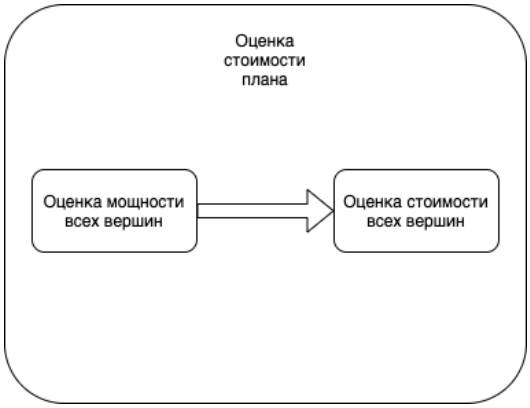


Рисунок 1 Схема оценки стоимости плана

Оценка стоимости всех вершин происходит по формуле (1),

$$cost = n_s c_s + n_r c_r + n_t c_t + n_i c_i + n_o c_o \quad (1)$$

в этой формуле n - мощности вершин, c - константные коэффициенты доступа к диску. Регулируя эти коэффициенты, можно добиваться более точной оценки стоимости плана. Однако на практике, значения по умолчанию этих коэффициентов обеспечивают достаточную точность предсказания. Поэтому подбор данных коэффициентов не является слабым местом этой схемы.

В исследовании [1] было показано как неправильная оценка, количества кортежей может влиять на вычисление стоимости плана.

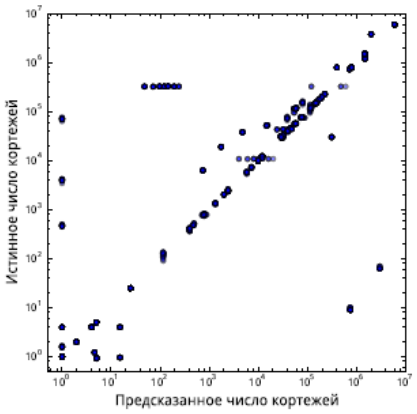


Рисунок 2а

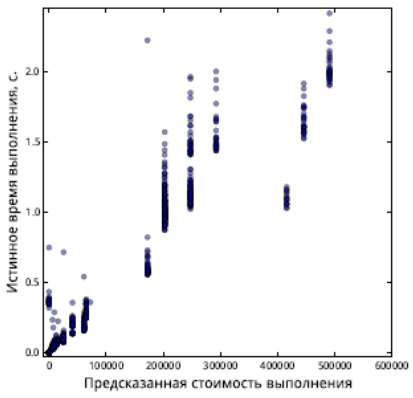


Рисунок 2б

На графике, который представлен на рисунке 4, расположены вершины некоторого плана запроса, который был выбран стандартными средствами СУБД PostgreSQL. Во всех вершинах, так же стандартными методами СУБД, было посчитано количество кортежей, и стоимость выполнения, далее также для всех вершин были записаны мощность и время выполнения после

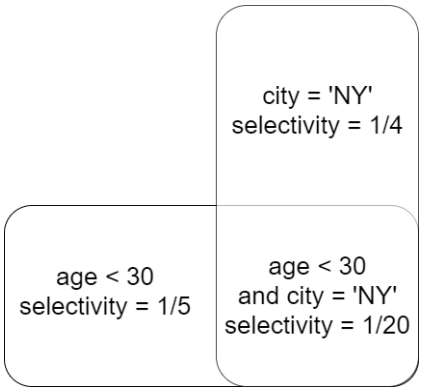
фактического выполнения запроса. На рисунке 2а изображен результат решения задачи вычисления мощности вершин, а на 2б изображены вершины, для которых точность оценки мощности отличалась не более чем на 10% от фактической. На левом графике масштаб осей логарифмический, из этого следует, что ошибка в данной задаче может быть в несколько порядков. На графике 2б видно, что в случае верной оценки мощности, стоимостная функция точно оценивает стоимость плана.

Из этого можно сделать вывод, что проблемным местом текущего стоимостного метода выбора плана запроса является оценка мощности плана.

Анализ предлагаемого решения

Неправильная оценка мощности плана происходит по причине неверного подсчета совместной выборочности всех условий запроса. Эта проблема присутствуем во многих современных реляционных СУБД, и в частности PostgreSql. Она связана с тем, что условия в запросе считаются независимыми, даже в тех случаях, когда условия определенно зависят друг от друга.

К примеру, мы хотим получить информацию о всех пользователях младше 30 лет и живущих в городе Нью Йорк. Совместная выборочность двух условий, будет равна произведению выборочностей двух условий в отдельности. Но бывают ситуации, когда условия не дополняют, а исключают друг друга и в таких случаях, где мы хотим получить всех женатых пользователей младше 10 лет, совместная выборочность логично равняется 0, однако на практике совместная выборочность также будет вычисляться как произведение выборочности отдельных условий.



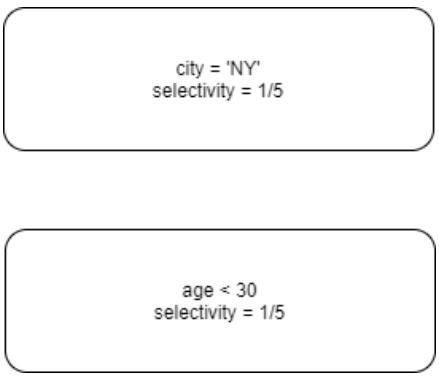


Рисунок 3 Рисунок 4

В данной работе предлагается улучшать предсказание мощности вершин при помощи методов машинного обучения с использованием фактической статистики выполнения запроса.

Схема разработанной системы

Исходными данными, в соответствии с рисунком 5 являются дерево разбора запроса, получаемое после этапа семантического анализа, а также исходный текст запроса.

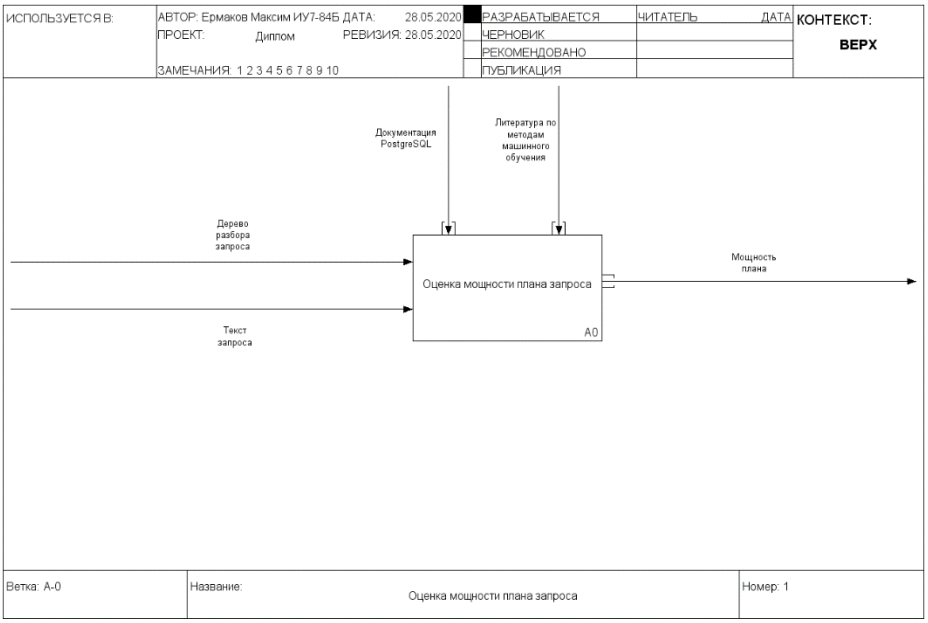


Рисунок 5 Диаграмма схемы работы

После декомпозиции данной задачи было выделено два основных этапа. Сначала формируется признаковое пространство необходимое для предсказания мощности вершины плана одним из методов машинного обучения. Далее собственно сам алгоритм машинного обучения предсказывает значение мощности. Данная декомпозиция указана на рисунке 6.

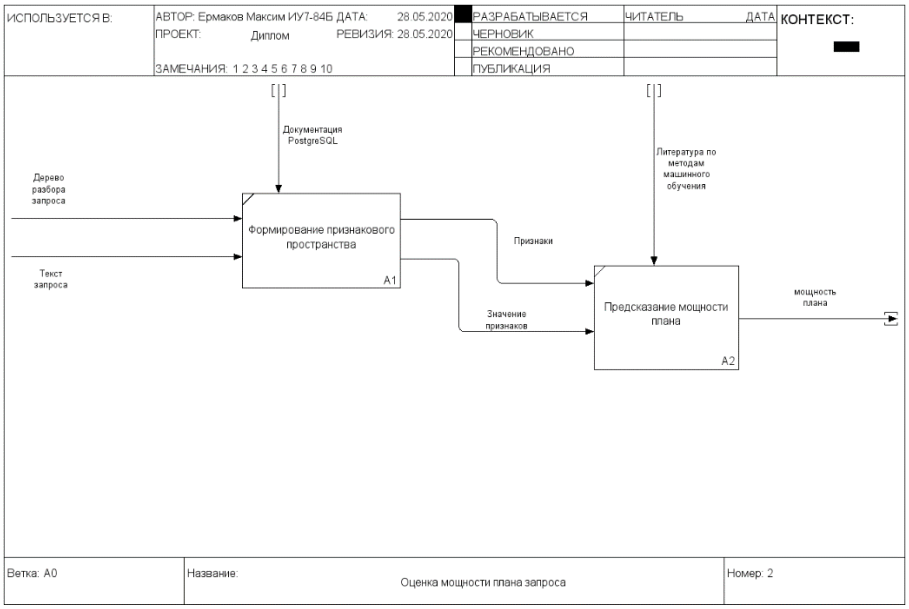


Рисунок 6 Декомпозиция задачи

Формирование признакового пространства

Главная идея данной работы – использовать статистику, которую выдает СУБД после фактического выполнения запроса, а именно фактическую мощность каждой вершины плана. Дерево плана может быть произвольной сложности с сложными условиями в каждой из вершин этого плана. Ни один метод машинного обучения напрямую не может работать с подобными структурами данных. Для того чтобы предсказывать мощность вершины дерева плана необходимо знать какие отношения она объединяет и при каких условиях. Количество комбинаций различных условий очень велико, для решения этой проблемы было реализовано следующее решение: два условия считаются равными в случае, если они отличаются только константами.

На вход методу машинного обучения подается набор признаков и их значений. В данном случае признаками будут выступать условия запроса с удаленными из них константными значениями. Однако в константных значениях заключена информация о выборочности этих условий, для того чтобы сохранить эту информацию в качестве значений признаков было принято решение использовать выборочности каждого условия, которые подсчитаны по гистограммам средствами PostgreSQL, так как по выборочностям условий можно восстановить константные значения в запросе.

Важно отметить, что пространство признаков не конечномерно, но на практике во всех запросах всегда ограниченное число условий. Таким образом, объект машинного обучения – вершина плана запроса, признаки – условия в запросе, значения признаков – выборочности каждого условия, целевое значение для предсказания – мощность вершины плана

Выбор метода машинного обучения

Выбор делается из 4 методов: линейная регрессия, нейронные сети, градиентный бустинг и метод к-ближайших соседей.

В качестве обучающей выборки для каждого метода используется таблица следующего вида.

Таблица 2.

обучающая выборка

Мощность	Признак #1	Признак #1	Признак #1
100	0.1	0.6	0.34
200	0.65	0.72	0.53
300	0.86	0.23	0.61

Данная таблица характеризует условия и фактическую мощность вершины, получаемую после выполнения запроса, значение каждого признака есть выборочность условия, посчитанная по гистограммам средствами PostgreSQL. Каждая строка в таблице соответствует одному выполнению плана с вершиной с такими условиями. Выборочности одно и того же условия (признака) различаются в каждой строке, это обусловлено тем, что из каждого условия удаляются константы и условия, которые отличаются только константами считаются эквивалентными, однако, чтобы учесть эти константы, в качестве значений признака используются выборочности условий, которые напрямую зависят от этих констант.

Таким образом для каждого набора условий существует свое признаковое пространство.

Линейная регрессия

Линейная регрессия [4] выглядит следующим образом.

$$C = \sum_j \alpha * s_j$$

В качестве метода обучения линейной регрессии используется метод стохастического градиентного спуска. Сначала инициализируем вектор α нулями после чего вычисляем другие параметры модели по указанной формуле:

$$\alpha_t = \alpha_{t-1} - \eta_t \nabla Q(\alpha, \{x_i\})$$

Функционал ошибки $Q(\alpha, X)$ на выборке X записывается в матричном виде, как квадрат отклонения вектора прогноза от вектора истинных значений.

$$Q(\alpha, X) = \frac{1}{l} ||X\alpha - y||^2$$

Вектор градиента:

$$\nabla Q(\alpha, X) = \frac{2}{l} X^T (X\,\alpha - y)$$

Распишем j-ую компоненту этого вектора:

$$\frac{\partial Q}{\partial \alpha_j} = \frac{2}{l} \sum_{i=1}^{l} x_i^j (\langle \alpha, x_i \rangle - y_i)$$

Метод k-ближайших соседей

В данной работе рассматривается взвешенная модификация метода k-ближайших соседей для задачи регрессии. Соседями считаются строки из матрицы обучающей выборки. Расстояние между соседями вычисляется как

$$d = \sqrt{(a_1 - b_1)^2 + \cdots + (a_n - b_n)^2}$$

где

a – вектор значений признаков соседа

b – вектор значений признаков вершины плана

Веса соседей вычисляются как функция от расстояния:

$$w(d) = \frac{1}{0.001 + d}$$

Итоговое значение вычисляется как сумма весов соседей умноженных на значения соседей соответственно, то есть на значение мощности соседей. Эта сумма делится на обычную сумму весов всех соседей.

$$y = \frac{\sum_{i=1}^{l} w_i * y_i}{\sum_{i=1}^{l} w_i}$$

Минусом данного метода является то, что для каждого объекта (вершины плана) нужно хранить и накапливать обучающую выборку в оперативной памяти для улучшения результатов предсказания, однако спустя время обучающая выборка может занять все в место в памяти, что приведет к плачевным последствиям, так же это чем больше обучающая выборка, тем сложнее проводить поиск соседей.

Метод градиентного бустинга над решающими деревьями

Опишем алгоритм построения решающего дерева. Вершинами дерева являются признаки обучающей выборки, то есть в нашем случае условия в вершине плана запроса, в каждой вершине значение признака сравнивается с пороговом значением t, которое выбирается таким образом, чтобы разделить выборку попавшую в эту вершину дерева на две другие выборки, в данном случае все значения признака сортируются и t приравнивается медианному значению. Так как решается задача регрессии, то в листьях дерева высчитывается значение мощности для объекта, попавшего в соответствующий лист. Мощность объекта в листовой вершине высчитывается по следующей формуле:

$$y = \frac{\sum_{i}^{m} y_i}{|X_m|}$$

где $|X_m|$ – количество объектов в вершине m

Глубина дерева в методе принимается равной 8. Недостатком данного метода является то что его невозможно дообучить на новых данных. Для того чтобы учесть новые результаты выполнения плана запроса нужно переобучить модель заново, что занимает достаточно много времени.

Метод, основанный на нейронных сетях

Нейронные сети [2] могут быть полезны для решения проблемы так как у них есть возможность бесконечно уменьшать ошибку при подходящем размере скрытого слоя. В данной работе использовалась полносвязная нейронная сеть с одним скрытым слоем с 50 нейронами, в качестве функции активации используется сигмиоидная функция, а для обучения нейронной сети используется метод RMSProp [3], который является стохастической техникой обучения, поэтому будем хранить только 256 объектов для обучения

Обоснование выбора метода машинного обучения

Для обоснования выбранного метода был проведен вычислительный эксперимент. Были реализованы все 4 метода машинного обучения. Итоговая программа тестировалась с каждым из методов. Целью эксперимента было выяснить насколько хорошо каждый метод предсказывает мощность вершин. База данных для эксперимента состояла из 1 таблицы на 15 столбцов и 50 000 строк. Были сгенерированы запросы следующего вида:

$$\texttt{SELECT * FROM X WHERE}$$

$$a_1 < x_{c1} \ \texttt{AND} \ x_{c1} < b_1$$

$$\texttt{AND} \ a_2 < x_{c2} \ \texttt{AND} \ x_{c2} < b_2$$

$$\dots$$

$$\texttt{AND} \ a_{15} < x_{c15} \ \texttt{AND} \ x_{c15} < b_{15}$$

То есть генерируются похожие по структуре запросы, которые отличаются только константами. Константы запроса удовлетворяют условию

$$0 < a_i < b_i < 100$$

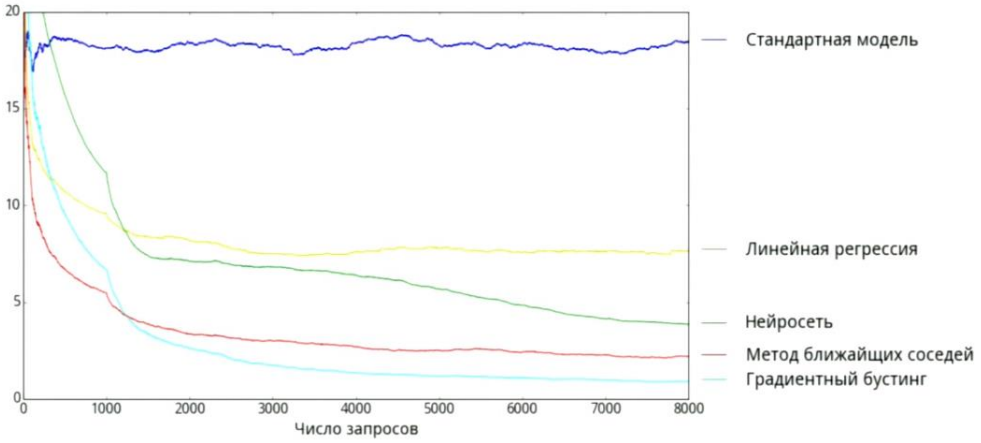


Рисунок 7 Результаты эксперимента

На графике на рисунке 7 по оси Y отложена ошибка предсказания. Как мы видим из графика результата эксперимента, меньше всего ошибку предсказания допускает метод градиентного бустинга, однако метод k-ближайших соседей за малое число запросов начинает показывать хорошую точность, это говорит о быстрой обучаемости метода, что на практике будет иметь большее значение. Таким образом в данной работе в качестве метода машинного обучения был выбран метод k-ближайших соседей.

Описания работы программного комплекса

На вход подается дерево плана запроса и текст запроса. Производится обход дерева плана. При посещении каждой вершины считается хеш от текста запроса сконкатенированного с условиями в этой вершине с удаленными константами. Данный хеш является идентификатором набора условий в вершине для этого запроса, ему соответствует матрица признаков и их значений, вектор реальных мощностей, полученных из предыдущих запросов.

Таким образом, при посещении вершины, по хешу извлекается нужное признаковое пространство, которое используется методом k-ближайших соседей для предсказания мощности вершины. Каждому набору условий в запросе, соответствует свое признаковое пространство. После исполнения запроса для каждой вершины в ее признаковое пространство будет дописан новый объект обучающей выборки – выборочности условий в вершине и реальное значение мощности вершины при таких выборочностях. Таким

образом происходит использование данных о мощностях вершин плана из предыдущего запроса. Важно отметить, что это позволяет не останавливать работу СУБД для того чтобы обучить модель на новых данных, новые данные, полученные из предыдущего запроса, автоматически будут использоваться при прогнозировании мощности следующего запроса. В случае если на вход программы подается абсолютно новый запрос, который отличается не только константами, то для него не будет сформированных признаковых пространств. Чтобы создать признаковые пространства для метода машинного обучения нужно несколько раз оценить мощность вершин с помощью стандартных методов PostgreSQL. После чего использовать полученные данные. Схема алгоритма предсказания мощности вершины представлена на рисунке 8.



Рисунок 8 Алгоритм обработки вершины плана

Алгоритм обучения

Процесс дообучения модели происходит с каждым новым выполненным запросом на фактической статистике выполнения этого запроса, для того чтобы избежать проблемы переполнения памяти обучающей выборкой было принято решение в обучающих выборках для всех вершин хранить не более 500 объектов, в случае если признаковое пространство еще не заполнено в него добавляется кортеж, состоящий из реального значения мощности вершины и

значения выборочностей, при которых такая мощность была получена. В случае если признаковое пространство уже заполнено, то нельзя просто так удалить любой элемент из признакового пространства. Находится ближайший сосед к новому элементу, и его место занимает новый элемент выборки, который мы хотим добавить.

Методика проведения экспериментов

В данном разделе были проведены вычислительные эксперименты и приведены их результаты, демонстрирующие качество работы системы.

Для проведения экспериментов в данной работе используется инструмент для тестирования производительности баз данных TPC-DS [5]. Данный тест является стандартом тестирования производительности, поэтому максимально приближен к реальной нагрузке на СУБД. Тест содержит в себе генератор базы данных и генератор 99 различных видов запросов, которые содержат в себе объединения отношений и варьируемые константы в условиях. Типы запросов отличаются по сложности, поэтому было принято решение разделить их на 5 групп по времени исполнения запроса со стандартной моделью оценки мощности запроса:

- Группа 1 - время исполнения меньше 1 секунды
- Группа 2 – время исполнения между 1 и 10 секундами
- Группа 3 – время исполнения между 10 и 100 секундами
- Группа 4 – время исполнения между 100 и 1000 секундами
- Группа 5 – время исполнения более 1000 секунд

Эксперимент проводился следующим образом, для каждого из 99 запросов было проведено 100 итераций выполнения запроса, таким образом была сформирована выборка соседей для метода k-ближайших соседей.

Ниже в таблице указаны результаты проведенного эксперимента, значение времен.

Таблица 3.

Результаты эксперимента

Группа запросов	Время выполнения запроса с стандартной моделью оценки мощности плана, в секундах	Время выполнения запроса с применением метода k-ближайших для оценки мощности плана, в секундах
1	50,197	398,239
2	533,349	612,02
3	5266,811	5384,691
4	56080,3	52516,715
5	153624,84	121238,063

Выводы из эксперимента

Для 1,2,3 групп запросов применение метода ухудшает время исполнения запроса. Это может объясняться тем, что в этих группах находятся запросы, с небольшим количеством узлов в планах, а как следствие такие запросы меньше зависят от ошибки предсказания мощности этих планов. Однако операция вычисления мощности в стандартной модели есть просто произведение выборочностей условий на количество обработанных кортежей, операция произведение нескольких чисел выполняется ощутимо быстрее чем решение задачи регрессии методом k-ближайших соседей на каждой вершине плана. В данном случае метод будет точно предсказывать мощность плана, но время оценки мощности будет перекрывать время, от точной оценки мощности.

На группах 4,5 в которых находятся запросы с большим количеством объединений отношений, то есть запросы, которые должны исполнятся долго, выигрыш по времени от более точной оценки мощности, а как следствие и стоимости выполнения плана перекрывает затраты на вычисление. При этом чем сложнее запросы, тем больше выигрыш по времени дает метод выбора

плана запроса, в котором мощности плана оцениваются при помощи метода k-ближайших соседей. Это происходит потому что для таких запросов строятся большие деревья планов, в которых ошибка в оценке мощности очень велика.

Литература

1. How good are query optimizers, really? / V. Leis, A. Gubichev, A. Mirchev et al // Proc. VLDB Endow. "– 2015. "– Nov… " Vol. 9, no. 3. " Pp. 204–215.
2. Нейронные сети. Полный курс /Саймон Хайкин. — 2-е изд. — Москва: Вильямс, 2016. — 1104с.
3. Методы оптимизации нейронных сетей [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://habr.com/ru/post/318970/
4. Линейная регрессия со стохастическим градиентным спуском [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://www.machinelearningmastery.ru/implement-linear-regression-stochastic-gradient-descent-scratch-python/, свободный
5. A Summary of TPC-DS [Электронный ресурс] /. — Электрон. журн. — Режим доступа: https://medium.com/hyrise/a-summary-of-tpc-ds-9fb5e7339a35
6. Кузнецов С.Д. Базы данных: языки и модели. М.: Бином-Пресс, 2008. 720 с.

Literature

1. How good are query optimizer, really? / V. Les, A. Gubichev, A. Mirchev et al / / Proc. VLDB Endow. "- 2015. "- Nov... "Vol. 9, no. 3." Pp. 204-215.
2. Neural networks. The complete course / Simon Haikin — - 2nd ed. - Moscow: Williams, 2016. - 1104s.
3. Methods of optimization of neural networks [Electronic resource]/. - Electron. journal. — Mode of access: https://habr.com/ru/post/318970/
4. Linear regression with stochastic gradient descent [Electronic resource]/. - Electron. journal. - Access mode: https://www.machinelearningmastery.ru/implement-linear-regression-stochastic-gradient-descent-scratch-python/, free

5. A Summary of TPC-DS [Electronic resource] /. - Electron. journal. — Mode of access: https://medium.com/hyrise/a-summary-of-tpc-ds-9fb5e7339a35
6. Kuznetsov S. D. Databases: languages and models. Moscow: Binom-Press, 2008. 720 p.