УДК: 004.052.2

Оценка планов выполнения SQL запросов для решения транспортных задач

Evaluation of execution plans of SQL query for solving transport problems

Дулин С.К., д.т.н., профессор, главный научный сотрудник, АО «НИИАС»; ИПИ ФИЦ ИУ РАН,
E-mail: skdulin@mail.ru, Москва, Россия
Dulin S.K., D.ofSci., Professor, Chief Researcher, JSC «NIIAS»;
Federal Research Center "Informatics and Management" of the RAS, E-mail: skdulin@mail.ru, Moscow, Russia

Рябцев А.Б., аспирант, Московский физико-технический институт (МФТИ),
E-mail: antr5@mail.ru, Москва, Россия
Ryabtsev A.B., Postgraduate student, Moscow Institute of Physics and Technology (MIPT),
E-mail: antr5@mail.ru, Moscow, Russia

Аннотация

Существующие подходы к проблеме поиска оптимального плана выполнения SQL запроса далеки от идеала. Учитывая многопараметрический характер транспортных задач, минимизация времени выполнения запроса может иметь решающее значение. Функция стоимости, которая каждому плану ставит в соответствие время его выполнения должна удовлетворять следующим требованиям: 1) отношение порядка стоимостей должно как можно больше совпадать с отношением порядка времени, 2) стоимость любого плана не может быть больше стоимости другого плана, полученного путём добавления операций соединений. В работе рассматривается задача оптимизации планов выполнения SQL запросов с помощью методов машинного обучения. В работе подробно описан традиционный подход к решению данной задачи, рассмотрены его недостатки. Также приведён анализ существующих методов машинного обучения, которые призваны устранить ряд недостатков традиционного оптимизатора. Рассмотрены их преимущества и недостатки.

Ключевые слова: транспорт, оптимальный план выполнения запроса, функция стоимости, методы машинного обучения, кардинальность таблицы.

Abstract

Existing approaches to the problem of finding a good SQL query execution plan are far from ideal. Given the multi-parameter nature of transport problems, the optimal plan that minimizes query execution time can be critical. The cost function that associates each plan with its execution time must satisfy the following requirements: 1) the order of cost ratio should match the order of time as much as possible, 2) the cost of any plan cannot be greater than the cost of another plan obtained by adding join operations. The paper considers the problem of optimizing SQL query execution plans using machine learning methods. The paper describes in detail the traditional approach to solving this problem, and considers its shortcomings. An analysis of existing machine learning methods is also given, which are designed to eliminate a number of shortcomings of the traditional optimizer. Their advantages and disadvantages are considered.

Keywords: trtransport, optimal query execution plan, cost function, machine learning methods, table cardinality.

»»

Введение

В современном мире транспорт является одним из ключевых факторов экономически эффективного функционирования промышленной отрасли. С точки зрения системного подхода — транспорт представляет сложную адаптивную экономическую систему, состоящую из взаимосвязанных в едином процессе транспортного логистического обслуживания региональных материальных и людских потоков. На транспорте можно выделить порядка 100-120 тыс. информационных объектов (железнодорожные и автобусные станции, морские и речные порты, аэропорты, транспортные предприятия в городах и поселках), около 20 тыс. железнодорожных станций, аэропортов, морских и речных портов, автобусных станций, более 25 тыс. железнодорожных участков, морских и речных путей, автомобильных дорог и авиалиний, порядка 5 тыс. наименований перевозочных средств транспорта. Все это предполагает фантастические объемы информационного контента транспортных систем и, соответственно, серьезные затраты на обслуживание запросов к транспортным базам данным, что делает особенно актуальным вопрос оптимизации реализуемых запросов. В ряде случаев имеет смысл обратить внимание на колоночные СУБД, демонстрирующие отличную функциональность, которая позволяет увеличивать производительность запросов и сжимать данные. Однако не следует рассматривать их как идеальное решение. Выигрыш будет получен лишь для соответствующих данных и SQL-запросов с определенными характеристиками. Тем самым, для транспортных информационных ресурсов необходимы исследования, связанные с оптимизацией запросов к базам данных, в частности, с анализом плана выполнения запроса.

Несмотря на то, что проблема поиска хорошего порядка соединения таблиц (плана выполнения запроса) – одна из наиболее изученных проблем в области баз данных, используемые на практики подходы далеки от идеала. При выполнении запроса соединение таблиц производится попарно, поэтому план выполнения запроса имеет структуру двоичного дерева, у которого листьями являются базовые отношения (таблицы), а промежуточными вершинами – операторы соединения. Планом выполнения запроса называют готовое дерево, предписывающее порядок соединения всех таблиц, участвующих в запросе. Подпланом же называют промежуточное состояние при построении плана – набор поддеревьев полного дерева (рисунок 1).

Под идеальным планом понимается план оптимальный, то есть тот, который выполняется быстрее других допустимых для выбранного запроса планов. Самый надёжный метод поиска оптимального плана – это полный перебор всех возможных планов и их выполнение с замером времени. План с самым малым временем выполнения есть оптимальный план. Понятно, что выполнять все возможные планы на практике никто не будет, но выбрать хороший план как-то всё-таки надо. Допустим существует функция, которая каждому плану соотносит его стоимость в условных единицах – в литературе эта функция называется функцией стоимости [5].

Идеальная функция стоимости устанавливала бы отношение порядка на всём пространстве возможных планов для всех возможных запросов. Но аналитически найти такую функцию пока никому не удалось. На практике же эту функцию аппроксимируют кусочной функцией на ручных правилах. В разных СУБД эти функции сделаны по-разному, но все они должны удовлетворять ряду требований: 1) отношение порядка стоимостей должно как можно больше совпадать с отношением порядка времени, так как выбирая самый дешёвый план мы надеемся выбрать самый быстрый; 2) стоимость любого плана не может быть больше стоимости другого плана, полученного из первого путём добавления соединений – в процессе выполнения более крупного плана, очевидно, будет выполнен и меньший план, являясь его подпланом, плюс какие-то ещё действия по присоединению.

Чтобы решить, что должно являться областью определения стоимостной функции, следует разобраться, что делает одни планы быстрыми, а другие медленными. В процессе выполнения плана происходит соединение таблиц. Соединение может производиться разными методами [8]. Например, самый примитивный метод – двойной цикл по ключам соединения двух таблиц для поиска совпадений. Временная сложность такого алгоритма $O(nm)$, где n – количество строк в одной таблице, а m – в другой. Более хитрый метод, называемый соединением с помощью хеша, строит по ключам одной таблицы хеш-таблицу, а по ключам второй таблицы выполняет поиск.

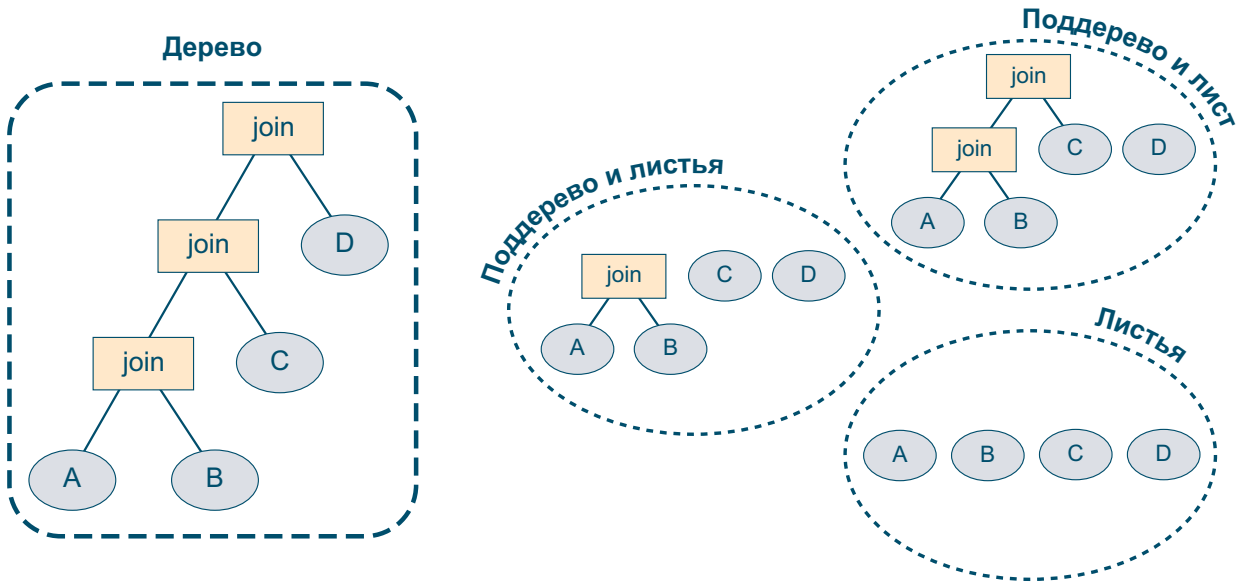
Временная сложность такого подхода в среднем $O(n+m)$. Чтобы понять какой метод эффективнее в том или ином случае, нужно знать размеры таблиц. Если две таблицы достаточно малы, то эффективнее будет воспользоваться примитивным способом соединения. Поэтому стоимостная функция должна учитывать как минимум размеры таблиц и способы их соединения. Размер таблицы традиционно называют кардинальностью [9].

Определение 1.1. Кардинальность таблицы – это общее количество строк, присутствующих в таблице в любой момент времени.

Определение 1.2. Селективность предиката – это доля строк таблицы, удовлетворяющих данному предикату.

Порядок соединения тоже важен, так как при определённой очередности соединений в промежуточных результатах можно получать таблицы поменьше, что благоприятно сказывается на общем времени выполнения. Кардинальности базовых отношений (таблиц) нам известны, чего нельзя сказать о размерах промежуточных результатов. Более того, наличие в запросе предикатов (фильтров) делает кардинальности промежуточных результатов соединений ещё более непредсказуемыми. Поэтому функция стоимости использует оценки кардинальности.

Как и многие оценки, они получаются на основе некоторых допущений, на практике зачастую оказывающихся ложными, что приводит к отклонениям оценок от реальных значений. Масштаб таких ошибок растёт с ростом числа таблиц, участвующих в запросе. Но на сегодняшний день в абсолютном большинстве СУБД оптимизаторы запросов работают именно так. Чтобы преодолеть »»



а) План

б) Подпланы

Рисунок 1. **План выполнения запроса и его подпланы**

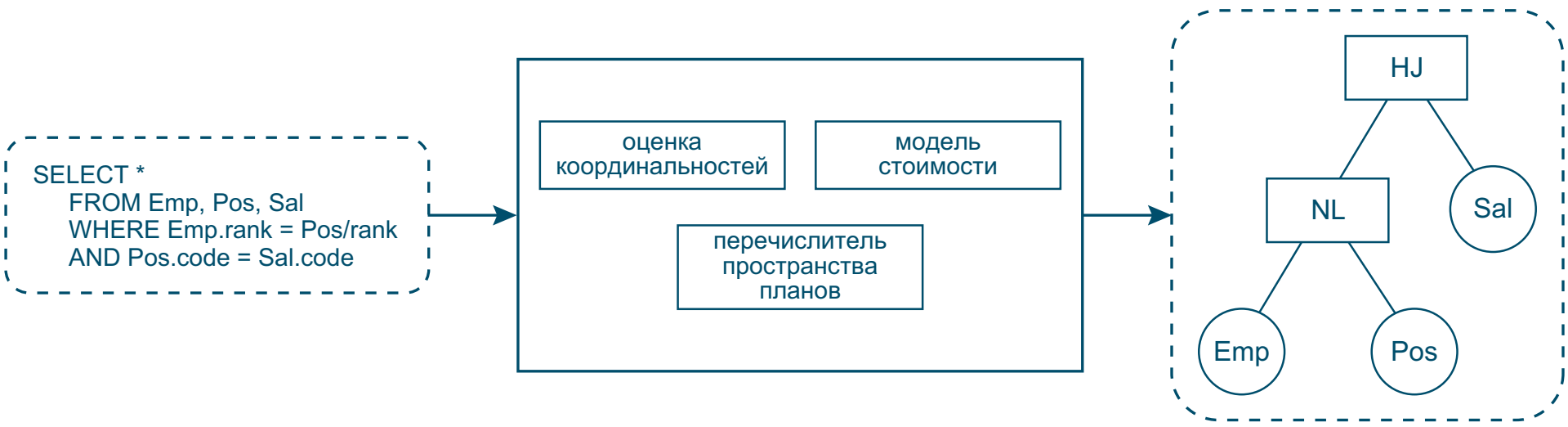


Рисунок 2. **Архитектура традиционного оптимизатора запросов**

имеющиеся слабые места традиционных подходов, можно прибегнуть к методам машинного обучения. Их можно применить как для более точных предсказаний кардинальности, так и для аппроксимации функции стоимости.

Традиционные методы

На рис. 2 показан классический [5] подход, основанный на получении оценок стоимости. Чтобы получить эффективный план выполнения запроса, оптимизатор запросов перечисляет некоторое подмножество допустимых порядков соединения, например, используя динамическое программирование. Используя оценки кардинальности в качестве основных входных данных, модель стоимости затем выбирает самую дешёвую альтернативу из семантически эквивалентных альтернатив планов. Теоретически, если оценки кардинальности и модель стоимости точны, эта архитектура получает оптимальный план запроса. На самом деле оценки кардинальности обычно вычисляются на основе упрощающих предположений, таких как равномерность и независимость данных в БД. В наборах данных реального мира эти предположения часто ошибочны оказываются ошибочны, что приводит к субоптимальным, а иногда и катастрофическим планам.

Модуль оценки кардинальности (CardEst) играет важную роль в оптимизации запросов. Он нацелена на получение оценок размеров результатов всех подпланов каждого запроса и руководит оптимизатором для выбора оптимальных операций соединения. Точность CardEst критически влияет на качество сгенерированных планов запросов. Из-за своей важной роли в СУБД CardEst широко изучается как академическим [12], так и индустриальным сообществом [26, 22, 19]. Современные СУБД с открытым исходным кодом и коммерческие СУБД в основном используют один из двух традиционных методов CardEst: гистограммы [3, 4, 7, 1, 2, 6] в PostgreSQL [24] и SQL Server [20], сэмплирование [10, 14, 15, 11, 18] в MySQL и MariaDB [23].

Подходы к оптимизации запросов на основе глубокого обучения

Совсем недавно сообщество баз данных начало исследовать возможности использования нейронных сетей для улучшения оптимизаторов запросов. Большая часть этих работ была сосредоточена на замене компонента оптимизатора обученными моделями. Например, DQ [16] и ReJOIN [17] используют обучение с подкреплением в сочетании с традиционными функциями стои- »»»

мости, созданными человеком, чтобы автоматически изучать стратегии поиска и исследовать пространство возможных порядков соединений. Эти статьи показывают, что выученные стратегии поиска могут превзойти обычные эвристические методы для данной функции стоимости. Более того, помимо традиционной функции стоимости, эти системы по-прежнему полагаются на эвристику для оценки кардинальности, выбора физического оператора и выбора индекса.

Другие подходы демонстрируют, как можно использовать машинное обучение для получения более точных оценок кардинальности. Кроме того, в отличие от выбора порядка соединения, выбор операторов соединения (например, хеш-соединение, соединение слиянием) и выбор индексов не могут быть полностью сведены к оценке количества элементов. SkinnerDB показала [28], что стратегии адаптивной обработки запросов могут выиграть от обучения с подкреплением, но для этого требуется специализированный (адаптивный) механизм выполнения запросов.

Подходы на основе более точной оценки кардинальности

Методы оценки кардинальности, основанные на машинном обучении, пытаются обучить модель для прямого сопоставления каждого запроса Q с его кардинальностью Card(T, Q). Методы оценки кардинальности по данным, основанные на машинном обучении, не зависят от запросов. Они рассматривают каждый кортеж в таблице T как точку, выбранную в соответствии с совместным распределением P(T(A)) = P(T (A1, A2, ..., An)). Пусть P(T (Q)) = = P(T (A1 ∈ R1, A2 ∈ R2, ..., An ∈ Rn)) — вероятность, определяемая областью Q. Тогда у нас есть Card(Q, T) = PR(Q) · |T|, так что задача оценки кардинальности может быть сведена к моделированию функции плотности вероятности P(T(A)) таблицы T.

AQO (Adaptive Query Optimizer) – встроенный в PostgreSQL подход AQO [13] усложняет стандартную формулу расчёта селективности: вводит для каждого простого условия свой собственный коэффициент. С помощью машинного обучения (используется регрессия методом ближайших соседей) AQO подбирает эти коэффициенты так, чтобы селективность, вычисленная по формуле, наилучшим образом соответствовала реальной селективности, которую AQO наблюдал ранее.

Среди всех подходов, что были рассмотрены в данной работе, этот подход самый простой как в плане реализации, так и в части идеи, стоящей за ним. Более того, этот подход уже внедрён в PostgreSQL и может быть включён в настройках системы. Однако в PostgreSQL по умолчанию используется три типа соединения отношений: соединение вложенным циклом, слиянием и с помощью хеширования. В OLAP системах, где пользователи имеют дело с большими аналитическими запросами, на практике используют только соединение с помощью хеширования, поскольку в среднем такие планы всегда работают быстрее. С ориентиром на OLAP системы, в рамках данной работы было проведено тестирование подхода для случая, когда включены только соединения с помощью хеширования. В такой конфигурации оптимизатора использование оценщика кардинальности AQO не дало ускорения времени выполнения запросов.

NeuroCard Подход построен на модели глубокой авторегрессии [25]. Он раскладывает совместную функцию плотности вероятности (ФПВ) по цепному правилу и моделирует каждую (условную) ФПВ параметрически с помощью 4-слойной нейронной сети. Стоит отметить, что исходный метод NeuroCard предназначен только для наборов данных с древовидной схемой соединения. В экспериментах данной работы с циклической схемой соединения производилось разделение схемы на несколько древовидных схем соединения и строилась отдельную модель NeuroCard для каждой схемы.

Метод точный и даёт желаемый результат. Но отличный прототип разбивается о продуктовую реальность. Число древовидных схем растёт экспоненциально с ростом числа таблиц в запросе, предсказание кардинальности делается с помощью нейросетевой модели, поэтому время вывода слишком велико для встраивания прототипа в оптимизатор промышленной СУБД.

Pessimistic CardEst [19] – альтернативный подход к получению оценок кардинальности. Здесь, в отличие от остальных подходов, предсказывается не кардинальность, а оценка сверху. И задача стоит не в том, чтобы как можно точнее предсказать значение, которое по факту может оказаться как меньше, так и больше фактического, а как можно сильнее улучшить верхнюю оценку, которая точно будет не меньше фактического значения.

За этим подходом стоит изящная теория – оценки выводятся через формулы условной энтропии, граф соединения представляется в виде гиперграфа для дальнейшей оптимизации вычислительной сложности, описано пространство для поиска компромисса между точностью и скоростью. Тем не менее подход всё-таки далёк от внедрения в оптимизатор СУБД, поскольку имеет ряд понятных ограничений, таких как невозможность работы с цикличными графами соединений и необходимость использовать в худшем случае линейную дополнительную память.

FLAT (fast, lightweight and accurate method) [29], основанный на сетях факторизации-разделения-суммы-произведения (FSPN), улучшается по сравнению с SPN за счет адаптивного разложения PT (A) в соответствии с уровнем зависимости атрибута. Добавляется узел факторизации PT на PT (W) · PT (H|W), где H и W являются сильно и слабо коррелированными атрибутами в T. PT (W) моделируется так же, как и SPN. PT (H|W) разбивается на небольшие ФПВ с помощью узлов разделения до тех пор, пока H не станет локально независимым от W. Затем многолистовой узел используется для непосредственного моделирования многомерной ФПВ PT(H). Подобно SPN, структура FSPN и вероятность запроса могут быть получены рекурсивно в нисходящем и восходящем порядке соответственно.

Идея разделения БД на кластеры сильно связанных между собой данных и слабо связанных с данными других кластеров позволяет использовать простые и, следовательно, быстрые модели машинного обучения, »»

поскольку зависимости, которые необходимо выучить после факторизации данных, достаточно хорошо аппроксимируются кусочно-линейными функциями. При изменениях данных в БД обновление структуры выполняется быстро. Этот подход является самым перспективным, поскольку он хитрее, чем AQO, и при этом уже успешно внедрён в одну коммерческую СУБД.

Подходы на основе замены традиционной функции стоимости на нейросетевую

Известно, что улучшение оценок кардинальности не гарантирует ускорение времени выполнения запроса. Повышение точности оценок кардинальности в 10 раз в экспериментах с MSCN (multi-set convolutional network) [21] привело к ускорению времени выполнения запросов всего на несколько процентов. А улучшение точности в двух других подходах в 3 и 2 раза соответственно вообще привело к увеличению времени выполнения запросов. При оптимизации запросов посредством улучшения оценок кардинальности всегда есть риск не прийти к желаемому, поскольку точность оценок не коррелирует с основной мерой качества – временем выполнения запросов. Помимо оценок кардинальности на итоговый результат также влияет используемая оптимизатором стоимостная функция, которая является своего рода эвристикой и разнится от СУБД к СУБД.

Понятные риски использования подхода на основе более точного предсказания кардинальности привели сообщество к идее оптимизации планов выполнения запросов напрямую, безотносительно оценок кардинальности. Идея этого класса подходов заключается в оптимизации стоимостной функции в обход расчёта оценок кардинальности. Здесь стали широко использоваться подходы на основе обучения с подкреплением (RL). Последовательное упорядочение соединений при построении плана выполнения запроса — это та же структура проблемы, которая лежит в основе RL. Можно использовать эту алгоритмическую связь, чтобы внедрить RL в традиционный оптимизатор запросов; везде, где используется алгоритм перечисления, стратегия, полученная из алгоритма RL, может быть легко применена.

Подход из статьи «Learning to Optimize Join Queries With Deep Reinforcement Learning»

DQN (Deep Q-Network – глубокая нейросеть, аппроксимирующая Q-функцию) формулирует оптимальное планирование как проблему прогнозирования [16]: учитывая затраты на ранее рассмотренные подпланы, какое одноэтапное решение, скорее всего, будет оптимальным? Говоря более конкретно, Q-learning устанавливает регрессию от решения объединить конкретную пару отношений к наблюдаемой выгоде от выполнения этого объединения по историческим данным (то есть влияние на окончательную стоимость всего плана запроса).

Подход из статьи «Neo: A Learned Query Optimizer» (Neural Optimizer) [27] строит обученный оптимизатор запросов, который обеспечивает аналогичную или улучшенную производительность по сравнению с современными коммерческими оптимизаторами (Oracle и Microsoft) на их собственных механизмах выполнения запросов. Neo учится принимать решения о порядке соединения и выборе оператора. Neo оптимизирует эти решения, используя обучение с учителем с петлёй "обратной связи" (постоянным дообучением), адаптируя себя к экземпляру базы данных пользователя и основывая своё решение на фактическом времени выполнения запроса.

Заключение

Подводя итоги, стоит отметить, что подходы на основе оценок кардинальности сами по себе не гарантируют ускорения запросов. Если у СУБД стоимостная функция подобрана не лучшим образом, улучшение оценок кардинальностей ничего не даст, и разработчикам придётся всё равно смотреть в сторону улучшения стоимостной функции. Подходы на основе аппроксимации функции стоимости в этом плане более перспективные. Однако, учитывая сложность задачи предсказания стоимости при ограничениях, характерных для транспортных задач, наиболее удачные подходы требуют колоссальных вычислительных мощностей, что делает невозможным их внедрение в СУБД. Оптимальным направлением в этих условиях представляется замена стоимостной функции не моделью машинного обучения, а просто более удачным вариантом кусочной функции на ручных правилах. ■

Список литературы

1. M Muralikrishna and David J DeWitt. Equi-depth multidimensional histograms. Proceedings of the 1988 ACM SIGMOD international conference on Management of data. 1988, pp.28–36.
2. P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie and Thomas G Price. Access path selection in a relational database management system. Readings in Artificial Intelligence and Databases. Elsevier, 1989, pp. 511–522.
3. Nicolas Bruno, Surajit Chaudhuri and Luis Gravano. STHoles: A multidimensional workloadaware histogram. Proceedings of the 2001 ACM SIGMOD international conference on Management of data. 2001, pp. 211–222.
4. Amol Deshpande, Minos Garofalakis and Rajeev Rastogi. Independence is good: Dependencybased histogram synopses for high-dimensional data. ACM SIGMOD Record 30.2 (2001), pp.199–210.
5. Abraham Silberschatz, Henry F Korth and Shashank Sudarshan. Database system concepts. Vol. 5. McGraw-Hill New York, 2002.
6. Hai Wang and Kenneth C Sevcik. A multi-dimensional histogram for selectivity estimation and fast approximate query answering. Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research. 2003, pp.328–342.
7. Dimitrios Gunopulos, George Kollios, Vassilis J Tsotras and Carlotta Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. VLDB Journal 14.2 (2005), pp. 137–154.
8. Gavin Powell. Beginning database design. John Wiley & Sons, 2006.
9. Hector Garcia-Molina. Database systems: the complete book. Pearson Education India, 2008.
10. Max Heimel, Martin Kiefer and Volker Markl. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. 2015, pp. 1477–1492.
11. Feifei Li, Bin Wu, Ke Yi and Zhuoyue Zhao. Wander join: Online aggregation via random walks. Proceedings of the 2016 International Conference on Management of Data. 2016, pp. 615–629.
12. Hazar Harmouch and Felix Naumann. Cardinality estimation: An experimental survey. Proceedings of the VLDB Endowment 11.4 (2017), pp. 499–512.
13. Oleg Ivanov and S Bartunov. Adaptive query optimization in PostgreSQL. PGCon 2017 Conference, Ottawa, Canada. 2017.
14. Martin Kiefer, Max Heimel, Sebastian Breß and Volker Markl. Estimating join selectivities using bandwidth-optimized kernel density models. Proceedings of the VLDB Endowment 10.13 (2017), pp. 2085–2096.
15. Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper and Thomas Neumann. Cardinality Estimation Done Right: Index-Based Join Sampling. Cidr. 2017.
16. Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph Hellerstein and Ion Stoica. Learning to optimize join queries with deep reinforcement learning. arXiv preprint arXiv:1808.03196 (2018).
17. Ryan Marcus and Olga Papaemmanouil. Deep reinforcement learning for join order enumeration. Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2018, pp. 1–4.
18. Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu and Ke Yi. Random sampling over joins revisited. Proceedings of the 2018 International Conference on Management of Data. 2018, pp. 1525–1539.
19. Walter Cai, Magdalena Balazinska and Dan Suciu. Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. Proceedings of the 2019 International Conference on Management of Data. 2019, pp. 18–35.
20. Pedro Lopes, Craig Guyer and Milener Gene. Sql docs: cardinality estimation (SQL Server). 2019.
21. Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich and Wolfgang Lehner. Cardinality estimation with local deep learning models. Proceedings of the second international workshop on exploiting artificial intelligence techniques for data management. 2019, pp. 1–8.
22. Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M Hellerstein, Sanjay Krishnan and Ion Stoica. Deep Unsupervised Cardinality Estimation. Proceedings of the VLDB Endowment 13.3 (2019).
23. MariaDB Server Documentation. Statistics for optimizing queries: InnoDB persistent statistics. 2020.
24. Postgresql Documentation. 12. 2020. Chapter 70.1. Row Estimation Examples. 2020.
25. Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen and Ion Stoica. NeuroCard: one cardinality estimator for all tables. Proceedings of the VLDB Endowment 14.1 (2020), pp. 61–73.
26. Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler et al. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. arXiv preprint arXiv:2109.05877 (2021).
27. Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil and Nesime Tatbul23. Neo: A Learned Query Optimizer. Proceedings of the VLDB Endowment 12.11 (2021).
28. Immanuel Trummer, Junxiong Wang, Ziyun Wei, Deepak Maram, Samuel Moseley, Saehan Jo, Joseph Antonakakis and Ankush Rayabhari. Skinnerdb: Regret-bounded query evaluation via reinforcement learning. ACM Transactions on Database Systems (TODS) 46.3 (2021), pp. 1– 45.
29. Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou and Bin Cui. FLAT: fast, lightweight and accurate method for cardinality estimation. Proceedings of the VLDB Endowment 14.9 (2021), pp. 1489–1502.