УДК 004.043

АНАЛИЗ ПОДХОДОВ К ОПТИМИЗАЦИИ ЗАПРОСОВ В АНАЛИТИЧЕСКИХ СУБД

Дулин Сергей Константинович[1,2],
д-р техн. наук, профессор,
e-mail: skdulin@mail.ru,

Рябцев Антон Борисович[3],
e-mail: antr5@mail.ru,
[1]*Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия*
[2]*Научно-исследовательский и проектно-конструкторский институт информатизации, автоматизации и связи на железнодорожном транспорте (АО НИИАС), г. Москва, Россия*
[3]*Московский физико-технический институт, г. Москва, Россия*

В работе рассматривается задача оптимизации планов выполнения аналитических SQL-запросов с помощью машинного обучения. На текущий момент ни одна из популярных систем управления базами данных (СУБД) не использует оптимизатор, основанный на машинном обучении. В то же время именно машинное обучение может решить известные проблемы стандартных оптимизаторов, используемых в большинстве современных СУБД. В данной работе детально рассмотрены недостатки существующих подходов к планированию аналитических SQL-запросов. Отмечено отсутствие исследований по применению известных подходов на основе машинного обучения к оптимизации SQL-запросов в аналитических СУБД. Показана актуальность решения проблемы оптимизации планов выполнения SQL-запросов для массово-параллельных колоночных СУБД. В работе приводится обоснование выбора конкретных методов на основе машинного обучения. Описаны произведённые модификации, в частности, предложен подход к выбору планов выполнения аналитических SQL-запросов на основе сбора обучающей выборки и выбора целевых значений. Рассмотрена проблема внедрения подхода на основе машинного обучения. Выделены возможные проблемы с обучением модели, её работой в принципиально новых, возникающих при обучении, ситуациях. Предложены способы их решения. Проведены оригинальные эксперименты с различными методами оптимизации SQL-запросов применительно к массово-параллельной колоночной СУБД. Показано, что предложенные модификации существующих решений значительно улучшают скорость выполнения запросов.

Ключевые слова: СУБД, аналитические SQL-запросы, план выполнения SQL-запроса, оценка стоимости, машинное обучение

ANALYSIS OF APPROACHES TO QUERY OPTIMIZATION IN ANALYTICAL DBMS

Dulin S.K.[1,2],
doctor of technical sciences, professor,
e-mail: skdulin@mail.ru,

Ryabtsev A.B.[3],
e-mail: antr5@mail.ru,
[1]*Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia*
[2]*Scientific Research and Design Institute of Informatization, Automation and Communication in Railway Transport (JSC NIIAS), Moscow, Russia*
[3]*Moscow Institute of Physics and Technology, Moscow, Russia*

The article describes the problem of optimizing the execution plans of analytical SQL queries using machine learning. Currently, none of the popular database management systems (DBMS) uses an optimizer based on machine learning. At the same time, it is machine learning that can solve the known problems of standard optimizers used in most modern DBMS. In this work, the disadvantages of existing approaches to planning analytical SQL queries are considered in detail. There is a lack of research on the use of well-known machine learning-based approaches to optimizing SQL queries in analytical databases. The relevance of solving the problem of optimizing SQL query execution plans for massively parallel column-based DBMS is shown. The paper provides a justification for the choice of specific methods based on machine learning. The modifications made are described, in particular, an approach to the selection of execution plans for analytical SQL queries based on the collection of a training sample and the selection of target values is proposed. The problem of implementing a machine learning-based approach is considered. Possible problems with the training of the model and its operation in fundamentally new situations arising during training are highlighted. The ways of their solution are proposed. Original experiments have been carried out with various methods of optimizing SQL queries in relation to a massively parallel column DBMS. It is shown that the proposed modifications of existing solutions significantly improve the speed of query execution.

Keywords: database management system, analytical SQL queries, SQL query execution plan, cost estimation, machine learning

DOI 10.21777/2500-2112-2023-3-73-80

Введение

Несмотря на то, что проблема поиска наилучшего порядка соединения таблиц (плана выполнения запроса) – одна из наиболее изученных проблем в области баз данных, используемые на практике подходы далеки от идеала. 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 называют промежуточное состояние при построении плана – набор поддеревьев полного дерева (рисунок 1).

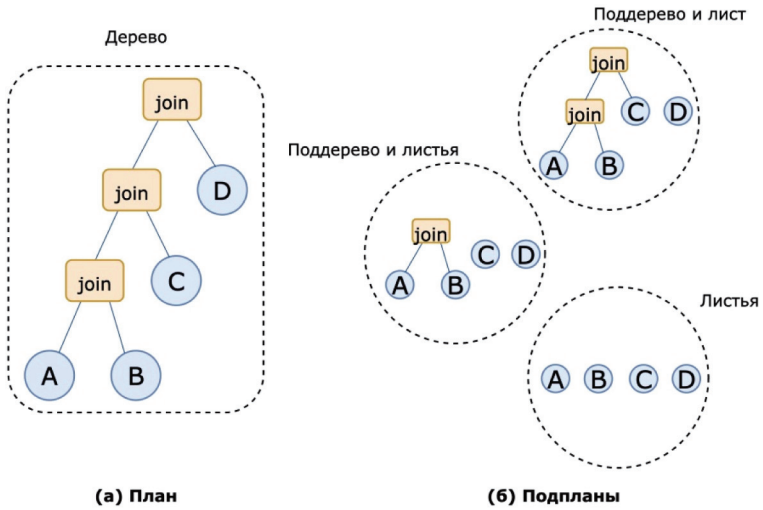

Рисунок 1 – План и его подпланы

Под идеальным планом понимается оптимальный план, который выполняется быстрее других допустимых для выбранного запроса планов. Самый надёжный метод поиска оптимального плана – это полный перебор всех планов и их выполнение с замером времени. Понятно, что выполнять все возможные планы на практике нецелесообразно, но выбрать хороший план необходимо. Допустим, существу-

ет функция, которая каждому плану соотносит его стоимость в условных единицах – в литературе эта функция называется функцией стоимости [1]. Идеальная функция стоимости устанавливает отношение порядка на всём пространстве возможных планов для всех возможных запросов. На практике эту функцию аппроксимируют кусочной функцией, подобранной вручную. В разных СУБД эти функции реализованы 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. Далее следует разобраться, что делает одни планы быстрыми, а другие медленными.

В процессе выполнения плана происходит соединение таблиц. Соединение может производиться разными методами [2]. Например, самый распространенный метод – двойной цикл по ключам соединения двух таблиц для поиска совпадений. Временная сложность такого алгоритма $O(n \times m)$, где n – количество строк в одной таблице, а m – в другой. Более сложный 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 предпочтительнее 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 [3].

Порядок соединения тоже важен, так как при определённой очередности соединений в промежуточных результатах можно получить таблицы поменьше, что благоприятно сказывается на общем времени выполнения. Кардинальности базовых отношений (таблиц) известны, чего нельзя сказать о размерах промежуточных результатов. 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 отношений, участвующих в запросе. На сегодняшний день в абсолютном большинстве СУБД оптимизаторы запросов работают именно так.

Чтобы преодолеть 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. Однако, в основном, все полученные на текущий момент результаты не столько производственные, сколько исследовательские. Во-первых, самые лучшие подходы в смысле точности получаемых оценок или в смысле времени выполнения планов запросов слишком «тяжеловесны» для встраивания в СУБД – ускорение выполнения запроса нивелируется ростом времени на построение плана. Во-вторых, эффективность подхода зависит также от самой СУБД, её структурных особенностей, и даже от выбранных настроек системы. В литературе нет примеров исследований подобных подходов для массово-параллельных колоночных СУБД [4]. При этом этот класс СУБД предпочтителен для решения аналитических задач, поскольку он обеспечивает эффективность выполнения больших аналитических запросов.

Решаемая в данной работе научно-практическая задача связана с использованием машинного обучения для оценки кардинальностей (CardEst), которая играет важную роль в оптимизации запросов, обеспечивая оценку размера результатов всех подпланов каждого запроса и выбор оптимальных операций соединения.

Цель работы – исследовать современные подходы машинного обучения для оценки кардинальности: FLAT (Fast, Lightweight, Accurate in esTimations) – разбиение на независимые части, построение отдельной модели вычисления кардинальностей для каждого случая; Pessimistic Cardinality Estimation – основанный на верхних оценках; NARU (Neural Relation Understanding) – авторегрессионная модель для одной таблицы; NeuroCard (Neural Cardinality estimator) – обобщение на несколько таблиц; PostgreSQL AQO (Adaptive Query Optimizer) – k ближайших соседей для поиска похожих прецедентов и расчёта оценки кардинальности по ним.

В данной работе исследовались несколько подходов машинного обучения для решения задачи оценки кардинальности в рамках применимости к массово-параллельным колоночным СУБД.

1. Реализация проведённых исследований

В качестве набора запросов использовались запросы к базе данных IMDB – Join Order Benchmark (JOB) [5]. Это набор так называемых «запросов реального мира». Такие наборы ценятся больше, чем наборы синтетических запросов, так как результаты, полученные на синтетических запросах, могут значимо отличаться от результатов на реальных запросах. Количество таблиц в запросах варьируются от 3 до 17. Всего в наборе около 100 запросов.

1.1 Машинное обучение для оценки кардинальности

В рамках данной работы проводились эксперименты на основе двух подходов: NeuroCard (Neural Network for Cardinality Estimation), AQO (Adaptive Query Optimizer).

1.1.1 NARU (Neural Relation Understanding) и NeuroCard (Neural Network for Cardinality Estimation)

Подход NeuroCard [6] – это развитие идеи NARU [7], имеющее возможность применяться для оценки кардинальности промежуточных результатов при соединении таблиц. Модель преобразуется в режиме обучения без учителя, исходя из данных в БД. При небольших изменениях в данных модель всё ещё может оставаться приемлемой, но, чтобы давать какие-либо гарантии, время, требующееся на обучение, должно быть кратно меньше времени, за которое в среднем данные меняются значительно. Понятно, что модель, требующая для обучения больше нескольких часов, это совершенно нереалистичный подход. Исходя из этого опыта, было принято решение рассмотреть подходы, которые уже встроены в СУБД с открытым исходным кодом. Одним из таких подходов является подход Adaptive Query Optimizer (AQO) [8], успешно встроенный в PostgreSQL.

1.1.2 PostgreSQL AQO (Adaptive Query Optimizer)

Авторы метода AQO утверждают, что для медленных запросов удалось добиться ускорения в 2 раза. Эксперименты авторов проводились с использованием PostgreSQL в стандартной конфигурации планировщика, то есть при построении плана были доступны как соединение с помощью хеша, так и соединение двойным циклом. В экспериментах данной работы в качестве СУБД использовалась массово-параллельная колоночная OLAP СУБД. На практике в таких системах возможность соединения двойным циклом отключена из практических соображений. Использование AQO в такой конфигурации планировщика не привело к изменениям планов из набора JOB IMDB, из чего вытекает гипотеза: AQO способен только исправлять «неудачные» NestLoop на HashJoin. Проверка этой гипотезы в рамках данной работы не проводилась.

1.2 Глубокое обучение для аппроксимации функции стоимости

В упомянутых выше статьях предлагается обучать модель на оценках стоимостей, а затем дообучать последний слой нейросети на времени выполнения запросов. На практике же использовать время выполнения запроса не получится, поскольку в зависимости от внешних факторов время выполнения для одного и того же плана может кратно различаться. Чтобы решить эту проблему, в данной работе было предложено использовать не время, а реальную стоимость плана, то есть стоимость, вычисленную на основе встроенной в традиционный оптимизатор функции, но только не на оценках кардинальности, а на их реальных значениях. Такой подход позволяет получить объективную, независимую от внешних факторов, характеристику, описывающую временную сложность выполнения плана.

1.2.1 DQN (Deep Q-Network) подход

Признаковое описание данных, представленное в статье DQN [9], не позволяет строить ветвистые планы. Легко доказать, что данный подход строит только левые-глубокие деревья. В качестве модели используется двухслойная полносвязная нейросеть (персептрон). Модель настраивается на предсказание Q-функции, которая вычисляет долгосрочный эффект от добавления в подплан конкретного соединения. Каждый тренировочный экземпляр – это кортеж (*state*, *action*, *cost*(*action*), *state'*). Нейросеть – это параметризованная модель для аппроксимации Q-функции.

Обозначим модель как Q_θ, Q_θ (f_{state},f_{action}) ≈Q(*state*, *action*),

где f_{state} – вектор, кодирующий состояние;

f_{action} – вектор, кодирующий действие;

θ – параметры модели, которые инициализируются методом Кайминга [10].

Для каждого кортежа из обучающей выборки можно вычислить целевое значение по формуле:

$$y_i = \text{cost}(action) + \min_{action'} Q_\theta(state', action'),$$

где *action*′ пробегает все возможные в состоянии *state*′ действия;

{y_i} могут быть использованы как целевые значения в задаче регрессии.

Функция ошибки вычисляется по формуле:

$$L(Q) = (1/N)\Sigma(y_i - Q_\theta(state, action))^2.$$

Параметры модели, аппроксимирующей Q-функцию, могут быть оптимизированы с помощью градиентных методов [11].

В описанном выше виде нейросеть не смогла обучиться. Для решения этой проблемы были предложены 3 модификации:

1. Добавление Layer Normalization [12] – техника в глубоком обучении, которая нормализует активацию нейронов в каждом слое сети. Она улучшает скорость и стабильность обучения, предотвращая взрывные или исчезающие градиенты.

2. Добавление к вектору входных данных числа, равного разнице количества таблиц в запросе и количества таблиц в текущем подплане.

3. Double Q-Learning [13] – вместо использования одной Q-функции для выбора действия и для оценки вознаграждения, Double Q-learning использует две Q-функции. Одна используется для выбора действия, а вторая – для оценки вознаграждения. Это помогает уменьшить смещение выборки, которое может возникнуть в обычном Q-learning.

С этими тремя модификациями тренировочный процесс начал сходиться. Однако лучший результат для набора запросов IMDB JOB уступал традиционному оптимизатору – общее время выполнения набора запросов было на 43 % больше, чем у традиционного оптимизатора (таблица 1). Одной из причин такого результата может служить ограничение на структуру планов, строящихся с применением этого подхода.

Таблица 1 – Анализ результатов подхода DQN

	Время выполнения 100 запросов	
	По планам классического оптимизатора	**По планам «умного» оптимизатора**
Вариант из оригинальной статьи	1.5 часа	7 часов
Вариант с рядом модификаций		2.15 часа

1.2.2 NEO (Neural Optimizer) подход

В подходе Neo не используется понятие *action*, подход оперирует только понятием *state*, оценивая перспективность каждого подплана вне зависимости от выбираемого на каждом шаге действия. Вместо остаточной стоимости модель предсказывает минимальную стоимость полного плана, который может быть достроен из текущего подплана.

У оригинального подхода (рисунок 2а) есть ряд недостатков. Во-первых, модель достигает приемлемого качества, когда её «опыт» достаточно велик. Экспериментально было выяснено, что в имеющейся постановке «достаточно велик» значит, что есть не менее 500 уникальных планов для каждого запроса. Это значит, что для сбора обучающей выборки нужно выполнить сотни неоптимальных планов для каждого запроса. Выполнение неоптимальных планов займёт неприемлемо много времени. На рисунке 2б представлена модификация архитектуры системы Neo, решающая эти проблемы. Вместо выполнения планов для них выполняется процедура EXPLAIN. Эта процедура использует традиционный оценщик кардинальности и стоимостную функцию, чтобы вычислить оценку стоимости плана. Выполняется эта операция моментально. За несколько секунд можно получить необходимое количество данных.

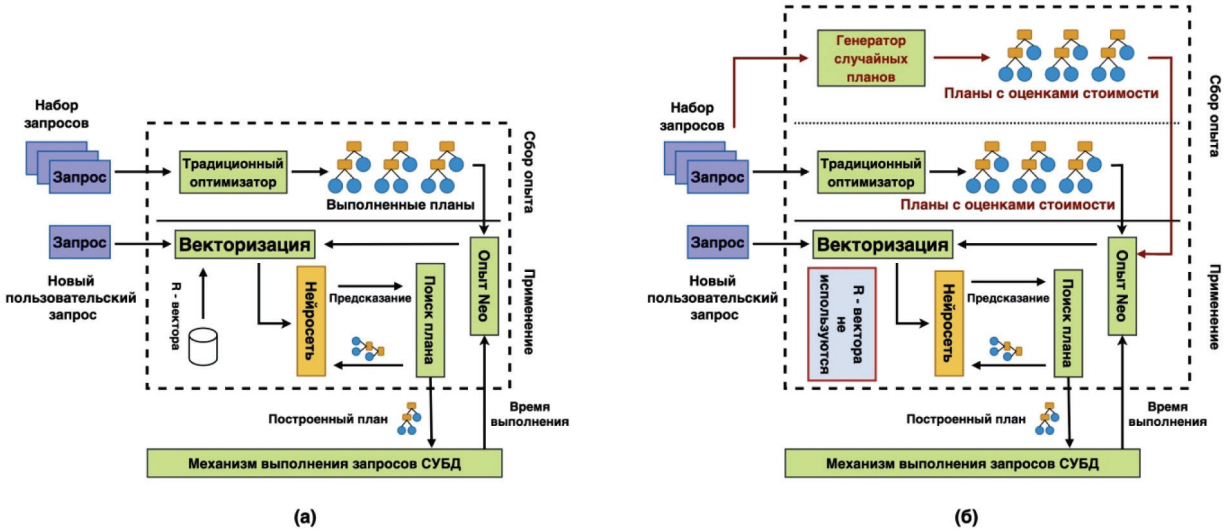

Рисунок 2 – Структура системы выполнения запросов: а) дизайн системы Neo в оригинале; б) модифицированный дизайн системы Neo – (модификации подсвечены бордовым цветом)

Так как входные данные представляют собой набор деревьев, а выходные – это одно число, на определённом этапе прохода через нейросеть требуется сделать объединение нескольких векторов в один. В оригинальной архитектуре это делается с помощью динамического объединения, например, с помощью выбора поэлементного максимума (Max Pooling) или среднего (Average Pooling) [14]. Это стандартный и простой подход, часто использующийся в задачах компьютерного зрения. Однако платой за простоту является потеря информации. Более продвинутым подходом, позволяющим преобразовать несколько векторов в один, является архитектура Transformer [15] с её механизмом внимания (Self-Attention). Архитектура нейросети из данной работы представлена на рисунке 3.

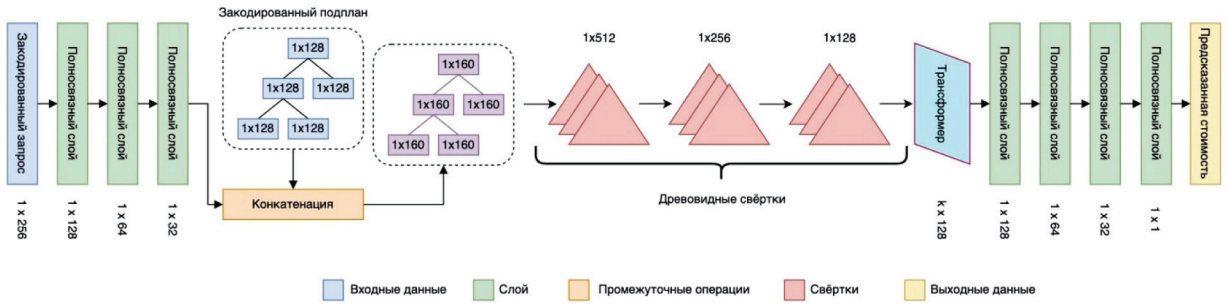

Рисунок 3 – Модификация архитектуры нейросети системы Neo

1.2.3 Получение оценок неопределённости предсказаний

Несмотря на преимущества нового подхода к составлению обучающей выборки, гарантий покрытия всей области определения нет. Более того, нет и гарантий отсутствия шума в самой обучающей выборке. Оба фактора могут приводить к ошибкам в предсказаниях нейросети. В данной задаче цена ошибки высока. Чтобы обезопасить систему от выбора потенциально неприемлемых подпланов, можно помимо предсказания стоимости оценивать также неопределённость этого предсказания. Если уровень неопределённости предсказания выше некоторого порога, этот подплан стоит исключить из рассмотрения вне зависимости от предсказанной ему стоимости.

1.2.4 Анализ результатов подхода Neo

Аналогично результатам с методом DQN, вариант подхода Neo из оригинальной статьи проигрывал традиционному оптимизатору. В рамках экспериментов с Neo использовались два разных режима СУБД: централизованный табличный режим и массово-параллельный колоночный режим. Для центра-

лизованного табличного режима вариант с модификациями смог построить планы, которые суммарно были на 40 % быстрее планов, строящихся традиционным оптимизатором. Однако для массово-параллельного колоночного режима «умный» оптимизатор построил планы на 30 % медленнее традиционных (таблица 2).

Таблица 2 – Анализ результатов подхода Neo

	Время выполнения 100 запросов			
	Табличный режим СУБД		**Колоночный режим СУБД**	
	Традиционный оптимизатор	**«Умный» оптимизатор**	**Традиционный оптимизатор**	**«Умный» оптимизатор**
Вариант из оригинальной статьи	2.5 часа	5 часов	1.5 часа	3 часа
Вариант с рядом модификаций		1.5 часа		2 часа

Из таблицы 2 следует, что предложенные модификации существующих решений значительно улучшают скорость выполнения запросов.

Заключение

Авторами работы поставлена задача оптимизации планов выполнения аналитических SQL-запросов с помощью машинного обучения. Проведены эксперименты по оценке эффективности методов машинного обучения для различных подходов. Осуществлен мониторинг и верификация полученных результатов применительно к массово-параллельной колоночной СУБД.

Подходы на основе оценок кардинальности показали себя недостаточно эффективными – одни не приводили к улучшениям, другие оказались слишком «тяжеловесны» для использования в СУБД. Подходы на основе аппроксимации функции стоимости также показали себя не лучшим образом. В оригинальном формате, как они были описаны в соответствующих статьях, они сильно проигрывают традиционному оптимизатору. После добавления ряда модификаций подход Neo демонстрирует лучшие характеристики по сравнению с традиционным оптимизатором на централизованном табличном режиме использовавшейся СУБД. Однако на массово-параллельном колоночном режиме традиционный оптимизатор оказался лучше. Причина кроется в самой стоимостной функции. В исследуемой СУБД стоимостная функция взята из централизованной СУБД с открытым исходным кодом, и никак не учитывает распределённый характер СУБД. Вследствие этого, время выполнения планов на незагруженной машине не коррелирует с реальными стоимостями этих планов, рассчитанных с помощью встроенной стоимостной функции и реальных значений кардинальности.

Продолжением исследования может быть не замена стоимостной функции моделью машинного обучения, а разработка новой, более приемлемой функции стоимости.

Список литературы

1. *Abraham Silberschatz, Henry F. Korth and Shashank Sudarshan*. Database system concepts. – New York: McGraw-Hill, 2002. – Vol. 5.
2. *Gavin Powell.* Beginning database design. – John Wiley & Sons, 2006. – 467 p.
3. *Hector Garcia-Molina.* Database systems: the complete book. – Pearson Education India, 2008. – 1119 p.
4. An Overview of MapD (Massively Parallel Database). – URL: https://smallake.kr/wp-content/uploads/2014/09/mapd_overview.pdf (date of application: 09/10/2023.). – Text: electronic.
5. *Shivnath Babu and Herodotos Herodotou*. Massively parallel databases and mapreduce systems // Foundations and Trends® in Databases. – 2013. – 107 p.
6. *Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper and Thomas Neumann.* How good are query optimizers, really? // Proceedings of the VLDB Endowment. – 2015. – Vol. 9, No. 3. – P. 204–215.
7. *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. – 2020. – Vol. 14, No. 1. – P. 61–73.

8. *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. – 2019. – Vol. 13, No. 3. – P. 279–292.
9. *Ivanov O. and Bartunov S.* Adaptive query optimization in PostgreSQL // PGCon 2017 Conference, Ottawa, Canada. – 2017.
10. *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.
11. *Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun.* Delving deep into rectifiers: Surpassing human-level performance on imagenet classification // Proceedings of the IEEE international conference on computer vision. – 2015. – P. 1026–1034.
12. *Гасников А.В.* Современные численные методы оптимизации. Метод универсального градиентного спуска. – М., 2018. – 272 с.
13. *Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao and Junyang Lin.* Understanding and improving layer normalization // Advances in Neural Information Processing Systems 32. – 2019.
14. *Hado Hasselt.* Double Q-learning // Advances in neural information processing systems 23. – 2010.
15. *Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin and Serge Belongie.* Kernel pooling for convolutional neural networks // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – P. 2921–2930.

References

1. *Abraham Silberschatz, Henry F. Korth and Shashank Sudarshan*. Database system concepts. – New York: McGraw-Hill, 2002. – Vol. 5.
2. *Gavin Powell.* Beginning database design. – John Wiley & Sons, 2006. – 467 p.
3. *Hector Garcia-Molina.* Database systems: the complete book. – Pearson Education India, 2008. – 1119 p.
4. An Overview of MapD (Massively Parallel Database). – URL: https://smallake.kr/wp-content/uploads/2014/09/mapd_overview.pdf (date of application: 09/10/2023.). – Text: electronic.
5. *Shivnath Babu and Herodotos Herodotou*. Massively parallel databases and mapreduce systems // Foundations and Trends® in Databases. – 2013. – 107 p.
6. *Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper and Thomas Neumann*. How good are query optimizers, really? // Proceedings of the VLDB Endowment. – 2015. – Vol. 9, No. 3. – P. 204–215.
7. *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. – 2020. – Vol. 14, No. 1. – P. 61–73.
8. *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. – 2019. – Vol. 13, No. 3. – P. 279–292.
9. *Ivanov O. and Bartunov S.* Adaptive query optimization in PostgreSQL // PGCon 2017 Conference, Ottawa, Canada. – 2017.
10. *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.
11. *Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun*. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification // Proceedings of the IEEE international conference on computer vision. – 2015. – P. 1026–1034.
12. *Gasnikov A.V.* Sovremennye chislennye metody optimizacii. Metod universal'nogo gradientnogo spuska. – M., 2018. – 272 s.
13. *Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao and Junyang Lin*. Understanding and improving layer normalization // Advances in Neural Information Processing Systems 32. – 2019.
14. *Hado Hasselt.* Double Q-learning // Advances in neural information processing systems 23. – 2010.
15. *Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin and Serge Belongie*. Kernel pooling for convolutional neural networks // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – P. 2921–2930.