УДК 004.8
DOI: 10.26102/2310-6018/2023.43.4.026

Об оценке стоимости запросов и разработке приложения для обработки ресурсоемких запросов

Е.В. Нурматова✉

МИРЭА – Российский технологический университет, Москва, Российская Федерация

Резюме. Быстрый рост объема хранимых данных обусловливает необходимость интеграции инструментов мониторинга, анализа и оптимизации запросов к базам данных для своевременного и корректного установления наиболее ресурсоемких из них. Эти обстоятельства определяют актуальность разработки программных инструментов для оценки причин медленных запросов с формированием различных вариантов оптимизации. В данной работе исследованы причины, влияющие на ресурсоемкость запросов выборки данных. Показаны основания для медленных запросов, такие как качество собранной статистики, использования индексов, указаний плану запроса, структуры запроса, корректность настройки параметров инициализации базы данных, а также возможные варианты решения выявленных причин. Исследование представляет интерес с точки зрения объяснения основ физических операций, обеспечиваемых подсистемой выполнения запросов, которая интерпретирует процедурный план исполнения запроса, оптимизируя стоимость. Для решения задачи ускорения медленных запросов на основе корректного процедурного плана предлагается разработка приложения, учитывающего состав анализируемых стоимостных, объемных и временных характеристик запросов для их оптимизации. Описаны результаты тестирования разработанной системы, позволяющей повысить производительность запросов. Оценивалась скорость выполнения запроса по следующим метрикам: операция доступа к данным, стоимость выражения, стоимость операции ввода/вывода, время процессора, затраченное время на обработку всей выборки. Выполнение экспериментов по оценке корректности выявления медленных запросов подтверждает целесообразность применения на практике результатов проведенных исследований и разработанного приложения.

Ключевые слова: реляционные системы, оптимизация запросов, select, эффективность индексов, статистика, оценка стоимости.

Для цитирования: Нурматова Е.В. Об оценке стоимости запросов и разработке приложения обработки ресурсоёмких запросов. *Моделирование, оптимизация и информационные технологии.* 2023;11(4). URL: https://moitvivt.ru/ru/journal/pdf?id=1462 DOI: 10.26102/2310-6018/2023.43.4.026

Query cost estimation and development of an application for processing resource-intensive queries

E.V. Nurmatova✉

MIREA – Russian Technical University, Moscow, the Russian Federation

Abstract. Rapid growth of stored data volume necessitates integration of tools for monitoring, analysis and optimization of database queries for timely and correct identification of the most resource-intensive queries. These circumstances determine the relevance of developing software tools for assessing the causes of slow queries with the formation of various optimization options. This paper examines the reasons influencing the resource-intensive queries of data sampling. The reasons for slow queries are shown, such as the quality of collected statistics, use of indexes, hints, query structure, correctness of database initialization parameter settings, as well as possible solutions to the identified causes. The study is interesting from the point of view of explaining the basics of the physical operations provided by the

query execution subsystem, which interprets the procedural plan of query execution to optimize the cost. To solve the problem of speeding up slow queries based on a correct procedural plan, we propose the development of an application that takes into account the composition of the analyzed cost, volume and time characteristics of queries to optimize them. The results of testing the developed system, which helps to improve the performance of queries, are described. The speed of query execution was evaluated by the following metrics: data access operation, expression cost, I/O operation cost, CPU time, time spent on processing the whole sample. Performing experiments to evaluate the correctness of identifying slow queries confirms the feasibility of applying in practice the results of the conducted research and the developed application.

Keywords: relational systems, query optimization, select, index efficiency, statistics, cost estimation.

For citation: Nurmatova E.V. Query cost estimation and development of an application for processing resource-intensive queries. *Modeling, Optimization and Information Technology*. 2023;11(4). URL: https://moitvivt.ru/ru/journal/pdf?id=1462 DOI: 10.26102/2310-6018/2023.43.4.026 (In Russ.).

Введение

Существующие программные средства позволяют получить предполагаемый и реальный планы выполнения запроса. Они формируют ряд показателей ресурсоемкости запроса, объединенных в одну функцию стоимости: циклы процессора, количество операций ввода-вывода (чтение и запись дисковых блоков), а также количество доступной памяти (влияет на соотношение циклов процессора и количество операций ввода-вывода) [1-3]. Среди них основными являются cost – стоимость выполнения и cardinality (или rows) – кардинальность. Менее эффективный запрос имеет довольно значительные их значения [4]. Качественный анализ эффективности запроса требует рассмотрения других дополнительных показателей, таких, как CPU cost – процессорная стоимость выполнения, IO cost – стоимость ввода-вывода, temp space – показатель использования дискового пространства [5-9].

В случае когда не хватает оперативной памяти для выполнения запроса (например, для проведения сортировок, агрегирования) и начинает использоваться дисковое пространство, можно с большой вероятностью констатировать неэффективный slow-запрос [10, 11].

Цель данной работы заключается в разработке приложения, оценивающего причины ресурсоемких запросов с формированием различных вариантов оптимизации.

Для начала выделим основные причины, которые могут влиять на скорость запроса [12-15] (Таблица 1):

Таблица 1 – Причины медленных запросов
Table 1 – Causes of slow queries

Основание	Вариант решения
Наличие и качество собранной статистики таблиц и индексов, используемых в запросах	– Подбор и следование стратегии индексирования с соблюдением критериев по производительности – Сбор и обновление статистики по индексам
Наличие и правильность применения индексов	– Создание / изменение индекса по данным, входящим в запрос в части where / having / group by / order by – Модификикация запроса с целью освобождения от блокирующих операций или создать индекс по функции, блокирующей индекс – Сбор статистики по индексу – Проверка наличия указаний, блокирующих работу индекса, например, NO_INDEX – Проверка настройки эффективных параметров инициализации БД – Проверка наличия более «сильных» индексов [16] – Проверка процента разреженности индекса

Таблица 1 (продолжение)
Table 1 (extended)

Основание	Вариант решения
Наличие и эффективность указаний (хинтов, hint) плану исполнения запросов	– Проверка наличия и правильности следования хинтов – Проверка отсутствия или изменения индекса, указанного в хинте
Качество построения структуры запроса	– Следование рекомендациям по написанию эффективного sql-кода – Перестройка логических условий части where / having в соответствии с правилами минимизации логических функций – Использование приемов улучшения запросов с соединениями данных из нескольких таблиц / БД. – Использование хинтов, например, для указания плану запроса целесообразности применения циклов при выполнении соединения данных: option (loop join)
Правильность настройки параметров инициализации БД	– Сократить задержки и уменьшить время, необходимое для увеличения файлов данных – Сократить время старта СУБД [14] – Сократить время при восстановлении БД

Материалы и методы

Длительность реализации запроса с точки зрения времени исполнения включает длительность декомпозиции запроса на подзапросы, обслуживания запроса на сервере, фиксации и передачи сообщения о результатах завершения на сервер узла администратора базы данных, снятия блокировок с записей базы данных.

Основными стоимостными характеристиками запроса являются стоимость выполнения запроса на заданном интервале времени C_s, стоимость передачи данных пользователю на удаленном узле C_c, стоимость хранения данных на сервере БД C_{srv}. Последний показатель определяется физическим объемом данных V_s и стоимостью хранения единицы объема данных (одной логической записи) на сервере k_s [16, 17]:

$$\mathrm{C}_s = V_s \cdot k_s. \quad (1)$$

Общая стоимость выполнения запроса:

$$C = \mathrm{C}_s + C_{srv} + C_c. \quad (2)$$

В состав анализируемых объемных характеристик входит также объем полезной информации в одном экземпляре записи, который определяется объемом входящих в него групп, объем информации, анализируемой подзапросом в соответствии с детерминированными условиями его выполнения, а также объем передаваемых данных на некотором узле вычислительной системы.

Таким образом, все возможные критерии эффективности, выбираемые с целью оптимизации, условно делятся на три больших класса: минимизации по временным, стоимостным и объемным характеристикам [18-20].

Данные критерии учитываются программой обработки slow-запросов, клиентский интерфейс которой включает разделы:

1. Модуль ввода и анализа запроса, планируемого к оптимизации.

2. Модуль построения подходящих вариантов оптимизации запроса, введенного на предыдущем этапе.

3. Модуль отображения итогового оптимизированного запроса с описанием совершенных действий в системе управления данными.

Для работы с приложением необходимо иметь как минимум один сервер и расположенную на нем одну базу данных.

Система проверяет возможность оптимизации полученного запроса по

следующим параметрам: обновление статистики; создание индексов полей; добавление указаний (hint); создание псевдонимов таблиц (alias); замена запроса на эквивалентное выражение с меньшей стоимостью; добавление оператора top; создание view. В качестве критерия оптимизации выступает минимальное время реакции, которое нужно для обработки и вывода первой строки данных. Такой критерий может быть использован для запросов, не содержащих агрегирующих функций и не требующих их упорядочивания. Информация о том, как распределены данные в таблицах, участвующих в запросах, предоставляется статистикой.

Поэтому первоначальная обработка запроса начинается с опроса по показателям обновления статистики. Фрагмент принятия решения относительно предлагаемого варианта оптимизации приведен на Рисунке 1, и далее показана функция просмотра статистики приложения:

```
def view_statistic(self, table_name, col_name):
        query = f"""SELECT DISTINCT
        OBJECT_NAME(s.[object_id]) AS TableName,
        c.name AS ColumnName,
        s.name AS StatName,
        s.[object_id],
        s.stats_id,
        sc.stats_column_id,
        sc.column_id,
        STATS_DATE(s.[object_id], s.stats_id) AS LastUpdated
        FROM sys.stats s JOIN sys.stats_columns sc
                        ON sc.[object_id] = s.[object_id] AND sc.stats_id =
s.stats_id
        JOIN sys.columns c ON c.[object_id] = sc.[object_id] AND c.column_id =
sc.column_id
        JOIN sys.partitions par ON par.[object_id] = s.[object_id]
        JOIN sys.objects obj ON par.[object_id] = obj.[object_id]
        WHERE OBJECTPROPERTY(s.[object_id],'IsUserTable') = 1
        --AND (s.auto_created = 1 OR s.user_created = 1)
        and OBJECT_NAME(s.[object_id]) = '{table_name}' and c.name =
'{col_name}';"""
        return self.get_cursor().execute(query)
```

Если меры по добавлению индексов и подсказок не приводят к желаемым показателям по времени реакции, система вызывает функцию подбора эквивалентного запроса.

Рассмотрим вариант оптимизации запроса, выбирающего лекарственные средства по заданному коду регистра лекарственных средств России (РЛС), точнее не входящие в него:

```
select * from drug_registry right join med on drug_registry.kodr=med.rg_med
WHERE kodr is null
```

Программа проверяет данные по сбору статистики таблиц и наличие для таблиц индексов по первичным ключам и по внешнему ключу. Также данный запрос можно выполнить по разным планам, в базе программы заложены достаточно громоздкие алгоритмы подстановки для разных видов выборки. В данном примере можно:

– выбрать все записи из таблиц `med`, `drug_registry`, соединить их по условию `drug_registry.kodr = med.rg_med`, затем для каждой полученной строки посчитать подзапрос и проверить второе условие;

– выбрать все записи из таблицы `med`, для каждой записи найти соответствие по условию `drug_registry.kodr = med.rg_med` в таблице `drug_registry` через индекс по

первичному ключу (на поле `kodr`), затем для каждой полученной строки посчитать подзапрос и проверить второе условие;

– выбрать все записи из таблицы `med`, каждую запись соединить по условию `drug_registry.kodr = med.rg_med` с таблицей `drug_registry` через индекс по первичному ключу. Предварительно посчитать подзапрос, добавив в него условие выборки `rg_med is null`, проверяя его для каждой строки.

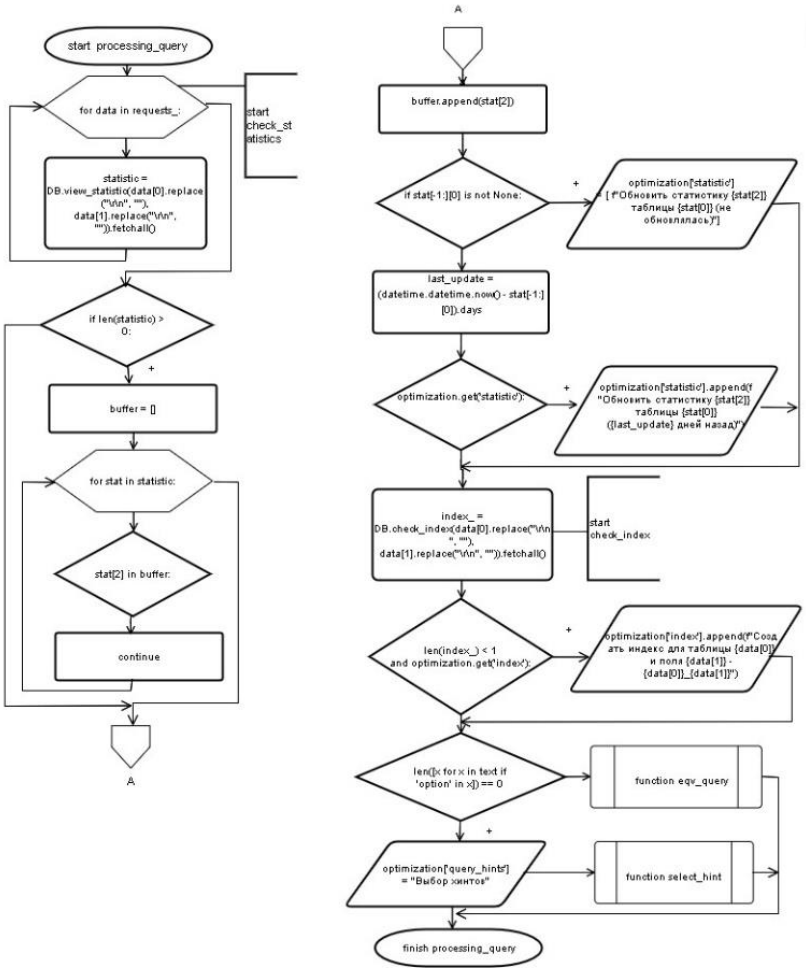

Рисунок 1 – Схема алгоритма выбора варианта оптимизации
Figure 1 – Schematic diagram of the algorithm for selecting the optimization variant

Первый план отличается от второго способом соединения исходных таблиц. Но декартово произведение всегда выполняется дольше, чем соединение через индекс. Не надо просматривать весь набор данных для поиска соответствия. Поэтому второй план всегда предпочтительнее первого. Третий план отличается предварительным вычислением подзапроса. Чем больше данных в таблице, тем больше разрыв по времени реакции в третьем запросе.

```
select * from med where rg_med not in (select kodr from drug_registry) or rg_med is null
```

Заложен еще вариант. Рассматриваемый запрос классифицируется как операция разности и может быть предложен вариант использования оператора EXCEPT, который извлечет все записи из первого набора данных, а затем удалит из результатов все записи из второго набора данных:

```
select * from med EXCEPT select * from med where rg_med in (select kodr from drug_registry)
```

Итак, после первоначальной обработки запроса будут предложены рекомендации по вариантам оптимизации, сделав выбор из которых, пользователь получит текст оптимизированного запроса.

Модуль обработки запроса осуществляет проверку корректности введенного запроса. В частности, происходит проверка существования используемых полей / таблиц, правильность следования конструкций запроса, обработка отправки пустой формы запроса. Текст, полученный из формы ввода, предварительно обрабатывается для определения возможных вариантов оптимизации. Данная страница приложения проиллюстрирована на Рисунке 2.

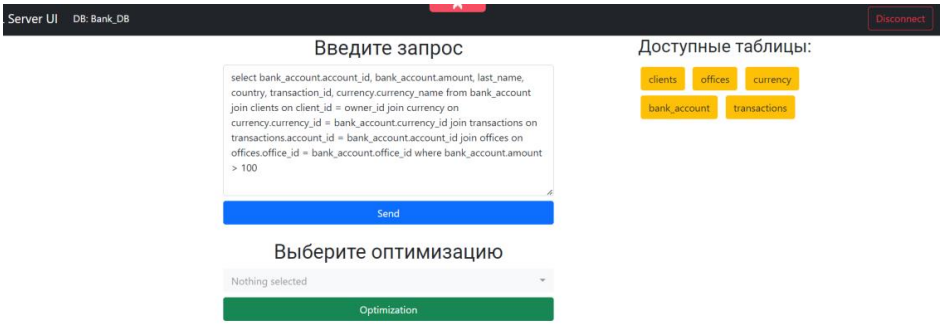

Рисунок 2 – Страница выбора вариантов оптимизации
Figure 2 – Optimization options selection page

Назначение основных функций модулей связи с данными и обработки запросов

Приложение имеет страницы подключения к выбранным наборам данных и непосредственной работы с select-запросом. На первой странице происходит подключение к необходимым серверу и базе данных, с которыми собирается работать пользователь. На второй странице происходит считывание введенного пользователем запроса, его передача в модуль обработки и вывод результата.

Таблица 2 – Описание модулей приложения
Table 2 – Description of the application modules

Наименование и параметры модуля	Назначение модуля приложения
new_connection(self, server_name, db_name)	подключение программы к серверу данных
view_statistic(self, table_name, col_name)	проверка на существование статистики и ее последнего обновления по заданным таблицам и полям
check_index(self, table_name, col_name)	проверка на существование индекса поля, которые пользователь ввел в запросе
check_columns(self, table_name, col_name)	проверка на существование в базе данных поля, которое пользователь использовал в запросе
update_statistic(self, table_name, st_name)	обновление статистики, выбранное пользователем в выпадающем списке предлагаемых вариантов оптимизации

Таблица 2 (продолжение)
Table 2 (extended)

Наименование и параметры модуля	Назначение модуля приложения
create_index(self, table_name, col_name, name)	создание индекса, которое выбрал пользователь в выпадающем списке предлагаемых вариантов оптимизации
create_view(self, text, name)	создание представления, которое выбрал пользователь в выпадающем списке предлагаемых вариантов оптимизации
get_requests()	функция, в которой происходят основные действия программы. В ней ведется поиск ключевых слов, запрос, подсчет слов «join», «as» и т. п., определяются запрашиваемые поля, таблицы, выводится решение по оптимизации

После того, как пользователь выбрал варианты оптимизации, модуль запускает процедуру обработки непосредственно данных. Сначала обновляется статистика при помощи хранимой процедуры с параметрами для запроса update statictis, создаются необходимые индексы, представления. Затем выполняется текстовое преобразование запроса: добавляются новые опции, относящиеся к выбранным раннее вариантам оптимизации. Например, добавляются операторы, хинты, имена таблиц.

После оптимизации модуль передает в веб-приложение информацию о совершенных им действиях и текст обновленного запроса.

Результаты

На Рисунке 3 перечислены сгенерированные приложением варианты оптимизации для запроса:

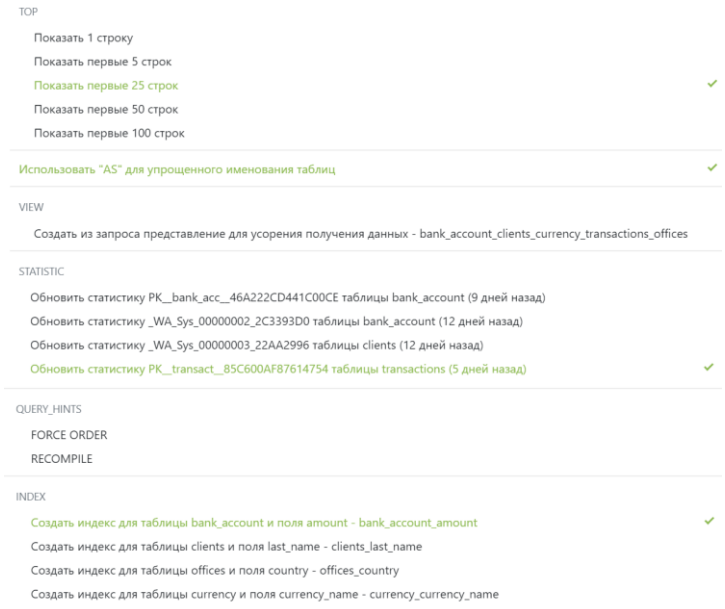

Рисунок 3 – Страница рекомендаций по оптимизации запросов
Figure 3 – Query optimization recommendations page

Пользователем были выбраны следующие оптимизации: обновление статистики PK__transact__85C600AF87614754, создание индекса для поля amount, показ первых 25-ти строк и упрощенное наименование таблиц. Итоговый select-запрос можно увидеть на Рисунке 4:

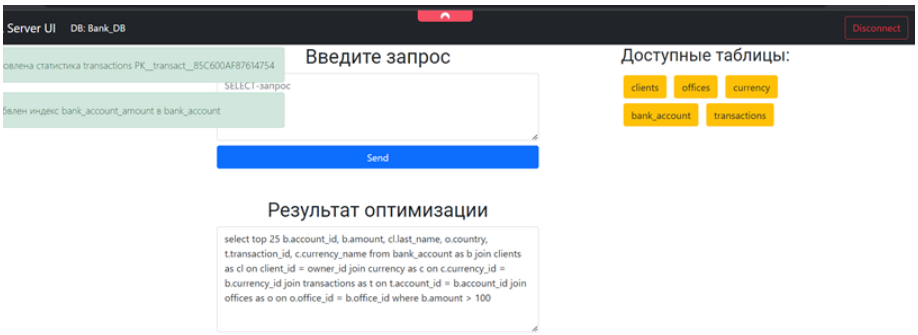

Рисунок 4 – Результат работы приложения
Figure 4 – Application result

Информацию о созданном индексе и обновленную статистику можно посмотреть в базе данных, с которой работало приложение.

Обсуждение

Для тестирования был использован запрос на выборку к базе данных, который был выполнен без применения индексации и с ним. Выборка производилась на основе схемы из трех таблиц: на 59 280 записей, на 3 724 654 записей и на 180 записей соответственно. Запрос включал операцию проекции на пять полей из этих таблиц, с двойным соединением таблиц и дополнительной горизонтальной селекцией.

В Таблице 2 сравнивается скорость выполнения запроса до и после применения индексации.

Таблица 2 – Скорость выполнения запроса
Table 2 – Request execution speed

Метрика	Без индексации	С индексацией
Операция доступа к данным	Сканирование кластерного индекса	Поиск по ключу
Стоимость выражения	12,013	0,089
Стоимость операции ввода / вывода	8,874	0,003
Время CPU (мс)	219	16
Затраченное время (мс)	195	63

Из данных таблицы видно, что индексация дает ощутимый прирост в производительности выполнения запроса. Нагрузка на процессор уменьшилась в 13.7 раз, а затраченное время в 3,1 раз.

Другой вариант тестирования модуля осуществлялся по запросу к секционированной таблице, состоящей из одной секции, секционированной по столбцу a. При этом был использован индекс по столбцу b, в запросе использована сортировка по данному полю. В результате, было рекомендовано применение хинта “ENABLE_QUERY_OPTIMIZER_HOTFIX”, который включает исправления оптимизатора в определенной БД.

Заключение

Основной результат работы состоит в разработке программного инструмента для оценки причин медленных запросов с формированием различных вариантов оптимизации и их последующей реализации.

Важно отметить, что оптимизация запроса предполагает намного более широкую работу, чем преобразование и оценка эквивалентности запросов, применение эффективных индексов, хинтов, обновление статистики и т. п. Несмотря на многие годы работы и исследований в этой области, существенные проблемы остаются открытыми [21].

Разработка предоставит возможность системам хранения и обработки данных, функционирующих с высокой нагрузкой, увеличить производительность, обеспечить оптимальное использование ресурсов, оптимизировать запросы и ускорить их выполнение.

СПИСОК ИСТОЧНИКОВ

1. Belattar S., Abdoun O., El khatir H. New learning approach for unsupervised neural networks model with application to agriculture field. *International Journal of Advanced Computer Science and Applications.* 2020;11(5):360–369.
2. Kim W. On optimizing an SQL-Like Nested Query. *ACM Transactions on Database Systems (TODS).* 1982;7(3):443-469.
3. Lukichev M., Barashev D. XML query algebra for cost-based optimization. *SYRCODIS*07 The Fourth Spring Young Researchers Colloquium on Databases and Information Systems*. 2007. URL: http://ceur-ws.org/Vol-256 (дата обращения: 21.09.2023).
4. Maher, M., Wang, J. Optimizing queries in extended relational databases. In: Ibrahim M., Küng J., Revell N. (eds) *Database and Expert Systems Applications. DEXA 2000. Lecture Notes in Computer Science.* 2000;1873. DOI: 10.1007/3-540-44469-6_36.
5. May N., Moerkotte G. *Normalization and translation of XQuery. Advanced Applications and Structures in XML Processing: Label Streams, Semantics Utilization and Data Query Technologies.* Hershey, Igi Global Publishing; 2010. 500 p. DOI: 10.4018/978-1-61520-727-5.
6. Кузнецов С.Д., Мендкович Н.А. Оптимизация конъюнктов условий в составе запросов. *Моделирование и анализ информационных систем*. 2011;18(3):144–154.
7. Nurmatova E.V., Gusev V.V., Kotliar V.V. Analysis of the features of the optimal logical structure of distributed databases. *Selected Papers of the 8th International Conference "Distributed Computing and Grid-technologies in Science and Education"*, 2018;2267:579–584. URL: https://ceur-ws.org/Vol-2267. (дата обращения: 21.09.2023).
8. Новиков Б.А., Горшкова Е.А., Графеева Н.Г. *Основы технологий баз данных.* 2-е изд. М.: ДМК Пресс; 2020. 582 с.
9. Борчук Л.Е. Совершенствование процесса настройки запросов пользователей на основе асимптотических оценок затрат ресурсов. *Информационные технологии*. 2008;6:6–11.
10. Борчук Л.Е., Кузьмин А.А. Оценка времени выполнения запроса в реляционной СУБД на основе асимптотических моделей затрат ресурсов. *Наукоемкие технологии.* 2008;4:61–64.
11. Домбровская Г., Новиков Б., Бейликова А. *Оптимизация запросов в PostgreSQL.* М.: ДМК Пресс; 2022. 278 с.
12. Зайцев Е.И., Нурматова Е.В. О подходе к управлению знаниями и разработке мультиагентной системы представления и обработки знаний. Russian Technological Journal. 2023;11(4):16–25. DOI:10.32362/2500-316X-2023-11-4-16-25.

13. Кузнецов С.Д., Мендкович Н.А. Новые алгоритмы лексической оптимизации запросов. *Модели и анализ информационных систем*. 2009;16(4):22–33.
14. Мендкович Н.А., Кузнецов С.Д. Оптимизация конъюнктов условий в составе запросов. *Модели и анализ информационных систем*. 2011;18(3):144–154.
15. Мендкович Н.А. Об эффективности минимизирующего подхода к оптимизации запросов. *Моделирование и анализ информационных систем*. 2016;23(2):153–163. DOI: 10.18255/1818-1015-2016-2-153-163.
16. Мосин С.В., Зыкин С.В. Кэширование запросов к реляционной базе данных с использованием областей истинности. *Моделирование и анализ информационных систем.* 2015;22(2):248–258. DOI: 10.18255/1818-1015-2015-2-248-258.
17. Нурматова Е.В. Анализ процедурного плана sql-запроса. *Опорный образовательный центр. Иннополис*. 2021;2:116–121.
18. Иванов О. Машинное обучение для планирования запросов. *Открытые системы. СУБД*. 2016. URL: https://www.osp.ru/os/2016/01/13048650 (дата обращения: 21.09.2023).
19. Пашинин О.В. Оптимизация запросов к базам данных. *Математические структуры и моделирование.* 2017;17:100–107.
20. Хайлан А.М., Польщиков К.А., Алгазали С.М.М. Обнаружение ресурсоемких запросов к базам данных на основе применения самоорганизующихся карт и нечеткого вывода. *Экономика. Информатика*. 2021:48(3):578–593. DOI: 10.52575/2687-0932-2021-48-3-578-593.
21. Wang S., Rundensteiner E.A., Mani M. Optimization of nested xquery expressions with orderby clauses. *Data & Knowledge Engineering*. 2007;60(2):303–325. DOI: 10.1016/j.datak.2006.03.004.

REFERENCES

1. Belattar S., Abdoun O., El khatir H. New learning approach for unsupervised neural networks model with application to agriculture field. International Journal of Advanced Computer Science and Applications. 2020;11(5):360–369.
2. Kim W. On optimizing an SQL-Like Nested Query. ACM Transactions on Database Systems (TODS). 1982;7(3):443-469.
3. Lukichev M., Barashev D. XML query algebra for cost-based optimization. SYRCODIS*07 The Fourth Spring Young Researchers Colloquium on Databases and Information Systems. 2007. URL: http://ceur-ws.org/Vol-256 (дата обращения: 21.09.2023).
4. Maher, M., Wang, J. Optimizing queries in extended relational databases. In: Ibrahim M., Küng J., Revell N. (eds) Database and Expert Systems Applications. DEXA 2000. Lecture Notes in Computer Science. 2000;1873. DOI: 10.1007/3-540-44469-6_36.
5. May N., Moerkotte G. Normalization and translation of XQuery. Advanced Applications and Structures in XML Processing: Label Streams, Semantics Utilization and Data Query Technologies. Hershey, Igi Global Publishing; 2010. 500 p. DOI: 10.4018/978-1-61520-727-5.
6. Kuznecov S.D., Mendkovich N.A. Optimizacija konjunktov uslovij v sostave zaprosov. *Modelirovanie i analiz informacionnyh sistem*. 2011;18(3):144–154.. (In Russ.).
7. Nurmatova E.V., Gusev V.V., Kotliar V.V. Analysis of the features of the optimal logical structure of distributed databases. *Selected Papers of the 8th International Conference "Distributed Computing and Grid-technologies in Science and Education"*, 2018;2267:579–584. URL: https://ceur-ws.org/Vol-2267. (дата обращения: 21.09.2023).

8. Novikov B.A., Gorshkova E.A., Grafeeva N.G. *Osnovy tehnologij baz dannyh.* 2-e izd. Moscow, DMK Press; 2020. 582 p. (In Russ.).
9. Borchuk L.E. Sovershenstvovanie processa nastrojki zaprosov pol'zovatelej na osnove asimptoticheskih ocenok zatrat resursov. *Informacionnye tehnologii*. 2008;6:6–11. (In Russ.).
10. Borchuk L.E., Kuz'min A.A. Ocenka vremeni vypolnenija zaprosa v reljacionnoj SUBD na osnove asimptoticheskih modelej zatrat resursov. *Naukoemkie tehnologii*. 2008;4:61–64. (In Russ.).
11. Dombrovskaja G., Novikov B., Bejlikova A. *Optimizacija zaprosov v PostgreSQL*. Moscow, DMK Press; 2022. 278 p. (In Russ.).
12. Zajcev E.I., Nurmatova E.V. O podhode k upravleniju znanijami i razrabotke mul'tiagentnoj sistemy predstavlenija i obrabotki znanij. *Russian Technological Journal*. 2023;11(4):16–25. DOI: 10.32362/2500-316X-2023-11-4-16-25. (In Russ.).
13. Kuznecov S.D., Mendkovich N.A. Novye algoritmy leksicheskoj optimizacii zaprosov. *Modeli i analiz informacionnyh sistem*. 2009;16(4):22–33. (In Russ.).
14. Mendkovich N.A., Kuznecov S.D. Optimizacija kon#junktov uslovij v sostave zaprosov. *Modeli i analiz informacionnyh sistem*. 2011;18(3):144–154. (In Russ.).
15. Mendkovich N.A. Ob jeffektivnosti minimizirujushhego podhoda k optimizacii zaprosov. *Modelirovanie i analiz informacionnyh sistem*. 2016;23(2):153–163. DOI: 10.18255/1818-1015-2016-2-153-163 (In Russ.).
16. Mosin S.V., Zykin S.V. Kjeshirovanie zaprosov k reljacionnoj baze dannyh s ispol'zovaniem oblastej istinnosti. *Modelirovanie i analiz informacionnyh sistem*. 2015;22(2):248–258. DOI: 10.18255/1818-1015-2015-2-248-258. (In Russ.).
17. Nurmatova E.V. Analiz procedurnogo plana sql-zaprosa. *Opornyj obrazovatel'nyj centr. Innopolis*. 2021;2:116–121. (In Russ.).
18. Ivanov O. Mashinnoe obuchenie dlja planirovanija zaprosov. Otkrytye sistemy. SUBD; 2016. URL: https://www.osp.ru/os/2016/01/13048650 (data obrashhenija: 21.09.2023) (In Russ.).
19. Pashinin O.V. Optimizacija zaprosov k bazam dannyh. *Matematicheskie struktury i modelirovanie.* 2017;17:100–107. (In Russ.).
20. Hajlan A.M., Pol'shhikov K.A., Algazali S.M.M. Obnaruzhenie resursoemkih zaprosov k bazam dannyh na osnove primenenija samoorganizujushhihsja kart i nechetkogo vyvoda. *Jekonomika. Informatika*. 2021:48(3):578–593. DOI: 10.52575/2687-0932-2021-48-3-578-593. (In Russ.).
21. Wang S., Rundensteiner E.A., Mani M. Optimization of nested xquery expressions with orderby clauses. *Data & Knowledge Engineering*. 2007;60(2):303–325. DOI: 10.1016/j.datak.2006.03.004.

ИНФОРМАЦИЯ ОБ АВТОРЕ / INFORMATION ABOUT THE AUTHOR

Нурматова Елена Вячеславовна, кандидат технических наук, доцент, доцент, РТУ МИРЭА, Москва, Российская Федерация.
e-mail: nurmatova@mirea.ru
ORCID: 0000-0001-8511-0978

Elena V. Nurmatova, Candidate of Technical Sciences, Associate Professor, Russian Technical University MIREA, Moscow, the Russian Federation.

Статья поступила в редакцию 26.10.2023; одобрена после рецензирования 04.12.2023; принята к публикации 20.12.2023.

The article was submitted 26.10.2023; approved after reviewing 04.12.2023; accepted for publication 20.12.2023.