УДК 004.22:004.27
DOI: 10.25559/SITITO.18.202203.558-565

Оригинальная статья

Создание частичного индексирования таблицы для оптимизации поисковых запросов

А. И. Миронов*, В. И. Мунерман
ФГБОУ ВО «Смоленский государственный университет», г. Смоленск, Российская Федерация
Адрес: 214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4
* 615153@mail.ru

Аннотация

В силу роста числа данных и роста разнообразия требований к их обработке, сейчас приходится отходить от обработки данных в момент запроса и все в большей мере перекладывать основную работу по его выполнению или выполнению основных его аспектов на заранее хранимые и подготовленные результаты. Во многом СУБД таким образом стараются решить проблемы производительности за счет увеличения расходов памяти, однако во многом необходимо задуматься уже об экономии последней, при этом желательно сохраняя результаты методов, основанных на подобном подходе – индексирование, хеширование, нейросетевые алгоритмы. В статье рассматривается метод повышения эффективности решения задач поиска для немалых таблиц. Предлагаемый метод основан на частичном индексировании, элементов, возле центров сближения и введения понятия метаданных для этих центров. Такая кластеризация с хранимыми метаданным для центров, около которых складываются очередные промежуточные узлы, позволяет снизить расходы памяти на индексацию, поскольку, во-первых, при таком подходе отсутствует необходимость вложенного индексирования, которые может привести к серьезным пространственным затратам. Во-вторых, такой подход может дать возможность использовать одно индексирование для разных комбинаций наличия столбцов в поисковом образе, не теряя при это большей части эффективности поиска при индексировании. Такое сочетание при правильном применении может позволить эффективно обрабатывать таблицы имеющие разные поисковые необходимости, по разным группам столбцов, для которых хранение индексации для каждого большого типа запроса или группы запросов может приводить закономерно к серьезным затратам на расход памяти, а также потерю производительности при работе с большими массивами памяти, которая тоже увеличивается далеко не линейно.

Ключевые слова: параллельное программирование, BigData, поиск сложных структур, кластеризация, базы данных, распределенные вычисления, частичное индексирование, карты Кохонена

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Миронов А. И., Мунерман В. И. Создание частичного индексирования таблицы для оптимизации поисковых запросов // Современные информационные технологии и ИТ-образование.2022.Т.18,№3.С.558-565.doi:https://doi.org/10.25559/SITITO.18.202203.558-565

Original article

Create Partial Table Indexing for Search Sources

A. I. Mironov*, V. I. Munerman
Smolensk State University, Smolensk, Russian Federation
Address: 4 Przhevalsky St., Smolensk 214000, Russian Federation
* 615153@mail.ru

Abstract

Due to the growing number of data and the growing variety of requirements for their processing, now we have to move away from data processing at the time of the request and increasingly shift the main work on its implementation or the implementation of its main aspects to pre-stored and prepared results. In many ways, DBMS thus try to solve performance problems by increasing memory consumption, but in many ways, it is necessary to think about saving the latter, while preferably preserving the results of methods based on a similar approach – indexing, hashing, neural network algorithms. The article discusses a method for improving the efficiency of solving search problems for large tables. The proposed method is based on partial indexing of elements near convergence centers and the introduction of the concept of metadata for these centers. Such clustering with stored metadata for the centers, near which the next intermediate nodes are formed, allows you to reduce the memory costs for indexing, because, firstly, with this approach there is no need for nested indexing, which can lead to serious spatial costs. Secondly, such an approach can make it possible to use one indexing for different combinations of the presence of columns in the search image, without losing most of the search efficiency during indexing. Such a combination, if used correctly, can make it possible to efficiently process tables with different search needs, for different groups of columns, for which storing indexing for each large type of query or group of queries can naturally lead to serious memory consumption costs as well as loss of performance when working with large arrays of memory, which also increases far from linearly.

Keywords: parallel programming, BigData, complex structure search, clustering, databases, distributed computing, partial indexing, Kohonen maps

Conflict of interests: The authors declare no conflict of interest.

For citation: Mironov A.I., Munerman V.I. Create Partial Table Indexing for Search Sources. *Modern Information Technologies and IT-Education.* 2022;18(3):558-565. doi: https://doi.org/10.25559/SITITO.18.202203.558-565

Введение

За последние годы в несколько раз вырос объем мультимедиа-данных. По некоторым подсчетам, количество информации в сети Интернет и на носителях-передатчиках на сегодняшний день составляет более 170 зеттабайт. При этом глубина или, иначе говоря, сложность оперируемых структур сильно усложняется со временем, особенно сейчас за счет повышения мощности носимых устройств, что привело в возможности работы с такими структурами у конечного пользователя. В таких условиях одной из самых распространенных задач остается задача поиска, которая с учетом новых вводных, становится куда сложнее. Нынешние поисковые запросы не заключаются в простейшем нахождении одного числа среди многих, они стали куда сложнее, в поисковых образах стало намного меньше определенности. Во-первых, поисковые запросы зачастую стали не точными, не рассчитанными на точное соответствие, сейчас многие из них устанавливают рамочные значения с тем, чтобы в результате получить наиболее релевантный результат. Во-вторых, данные с которыми оперируют нынешние решения становятся куда сложнее по своей структуре и СУБД или поисковым системам приходится работать с поисковыми образами, которые могут сильно отличаться от результата по характеристикам, при этом удовлетворяя поиску.

Существующие на сегодняшний день поисковые системы, осуществляющие отбор информации, не могут удовлетворить запросам новых требований в полной мере. Это происходит во-многом из-за того, что работа с информацией приобретает все более общный подход, в то время как алгоритмы, рассчитанные под простейшие данные, зачастую не меняются в этой концепции, и поэтому новые требования и новые архитектуры зачастую опираются на старые и не приспособленные под такие задачи инструменты [1-8].

В связи с этим возникает необходимость разработки таких алгоритмов, осуществляющих простейшие действия, как например поисковые запросы, которые изначально были бы рассчитаны на общный подход обращения с данными и при этом сохраняли бы производительность на достаточном уровне. Внедрение таких простейших алгоритмов, на которые как правило опирается немалая доля программных продуктов, иногда не непосредственно, позволит кратно увеличить производительность, причем чем сложнее изначальный программный продукт, тем более будет заметно преимущество перед использованием классических алгоритмов, что, необходимо учитывая складывающуюся тенденцию к усложнению программно-аппаратных комплексов.

В данной статье рассматривается возможность предложения метода, основанного на снижении размерности за счет карт Кохонена с применением частичной индексации для оптимизации задачи поисковых запросов по всем/нескольким столбцам.

Цель исследования

В настоящее время информация, подвергаемая обработке в СУБД имеет достаточно сложную типовую структуру. Сейчас трудно представить ситуацию, когда смысловая обработка ведется над обычными числами, тогда как все чаще обрабатываемые данные (представленные в табличном виде) олицетворяют собой сложные объекты, описываемые большим количеством разных характеристик. Так поиск по массивам информации – по частичным поисковым образам, в частности, когда известно множество, из условий которым должны удовлетворять значения характеристик искомого объекта становится все более обыденным и нормальным явлением. В это же время подобные задачи решают при помощи алгоритмов предполагавшими изначально работу с одним значением, что часто приводит к достаточно невысокой производительности. Целью данного исследования является попытка предложить алгоритм, основанный на индексировании, разбиении множества элементов вокруг центров и использовании вводимых метта-характеристик кластеризованной выборки для оптимизации работы частичного поиска для больших таблиц, с возможностью параллельной обработки. Данный алгоритм для кластеризации множеств предполагает алгоритм, основанный на упрощенном виде карт Кохонена и мер близости на пространстве элементов таблицы.

Кластеризация на основе карт Кохонена

Одной из частей предлагаемого алгоритма является упрощённая кластеризация по методу карт Кохонена. Опишем его вариант. Слой Кохонена состоит из некоторого количества n параллельно действующих линейных элементов. Все они имеют одинаковое число координат m – по количеству столбцов [1, С. 63]. Все элементы попарно сравниваются с центрами. После сравнения слоя линейных элементов ближайший из центров к отбирается для изменения. Если максимум одновременно достигается для нескольких центрах, то либо все принимаются для дальнейшего воздействия, либо только первый в списке (по соглашению). Количество центров равно количеству кластеров, среди которых происходит начальное распределение и последующее перераспределение обучающих примеров. Количество входных переменных равно числу столбцов, характеризующих элементы на основе которых происходит отнесение его к одному из кластеров.

После перечисления вводных понятий опишем примерную схему «обучения» классической самообучающейся карты Кохонена. Данную схему можно отнести к машинному обучения, однако она является довольно примитивным элементом такой классификации, притом, что подходит под поведением если представить линейные центры в качестве нейронов. Сети Кохонена довольно сильно используются в некоторых задачах кластеризации, для них же потребуются некоторые их характеристики и в данной работе.

Кластеризация включает следующие этапы:

1. Задание структуры «сети» (количества центров слоя глубины) (M).
2. Случайная инициализация весовых коэффициентов центров.
3. Подача случайного элемента текущего обучения и расчет расстояний по выбранной метрики или меры близости на базе таблицы, от входного вектора до центров всех будущих кластеров:

$$R_j = \sum |x_i - w_i|$$

По наименьшему из значений Rj будет выбран центр-победитель j, в наибольшей степени близкий по значениям с входным элементом. Для выбранного центра (и только для него) выполняется коррекция весовых коэффициентов – его сдвиг:

$$w_{i,j}(q+1) = w_{i,j}(q) + v\left(x_i - w_{i,j}(q)\right)$$

где *v* – коэффициент скорости обучения.

4. Цикл повторяется с шага 3 до выполнения одного или нескольких условий окончания: исчерпано заданное предельное количество эпох обучения; не произошло значимого изменения весовых коэффициентов в пределах заданной точности на протяжении последней эпохи обучения; исчерпано заданное предельное физическое время обучения.

Коэффициент скорости обучения может задаваться постоянным из пределов (0, 1] или переменным значением, постепенно уменьшающимся от эпохи к эпохе.

После такой процедуры каждый элемент множества будет сравнен с каждым центром на каждом уровне глубины. Номер того центра, что является ближайшим к входному объекту и является номером кластера, который соответствует входному элементу[1]. Центры кластера – случайные и сдвигаемые элементы которые являлись кластеризирующими в процессе. Учитывая тот факт, что эти объекты имеют ту же структуру что и объекты кластеризируемого множества, можно считать, что они так же принадлежат соответствующему кластеру – стоят в центре. Так же можно видеть, что кластеризация происходит при одних и тех же начальных весах одинаково. То есть если имеются начальные веса

$$\left\{\left\{x_1, x_2, \ldots x_k\right\}_{1,\ldots}, \left\{x_1, x_2, \ldots x_k\right\}\right\}$$

то кластеризация одного и того же множества при том же порядке элементов пройдёт таким же образом и с тем же результатом.

Предлагаемый способ индексирования

Рассмотрим подробно предлагаемую для оптимизации схему организации поиска и хранение метаданных для нее. В основе будут положены принципы сосредоточения на отсечении ненадлежащих(заведомо отдаленных по мерам схожести) имеющихся объектов внутри базы данных, а не на схождении к похожим образам. Так же необходимо учитывать тот факт, что классические индексы, построенные на нескольких ключах, заметно теряют в эффективности. Дело обстоит не столько в потере скорости при поиске объектов в базе, сколько в тратах на содержание индекса. В данный момент большинство индексов относительно нескольких столбцов заключается в их композиции. То есть множество сначала подвергается индексированию по одному полю, до предельного или константного значения мощности последних-листовых множеств, а затем уже каждое множество в листьях подвергается индексированию по другому столбцу и так далее. Надо ли говорить, что при такой схеме добавление новых данных становится чрезвычайно сложным, поскольку необходимо будет выполнить множество операций сравнения. В добавок к этому такая система плохо себя показывает при поиске не по той связке ключей, которая изначально подразумевалась для сравнения – причем достаточно изъятия из целевого сравнения хотя бы одного столбца чтобы заметно потерять в производительности. К тому же подобная организация системы катастрофически сильно влияет на объем занимаемой памяти, и если данные достаточно похожи, и, следовательно, деревья имеют высокую глубину, то в таком случае данные индексы столь же сильно сказываются на памяти, отводимой под базу (и это без учетов огромных таблиц журналирования) [9-15].

Подобную картину можно постараться исправить, несколько потеряв в скорости поиска – в каких-то моментах, однако сильно облегчив периферийное обслуживание базы данных[2]. Дальнейший алгоритм будет представлен на концептуальном примере для простоты изложения. Предположим, что у нас существует таблицы с достаточным количеством столбцов (очевидно, что для малого их числа достаточно стандартных индексов) и немалым количеством строк, для целей оптимизации поиска и сравнения.

Table Header															
Column A	Column B Column B Column B	Column C	Column D	Column E	Column F	Column G Column G	Column H	Column I	Column J	Column K	Column L	Column M	Column N	Column O	Column P
Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here	Column text here Column text here

Р и с. 1. Пример таблиц

F i g. 1. Table Example

[1] Ильин П. Л., Мунерман В. И. Рекурсивное вычисление детерминанта многомерной матрицы // Системы компьютерной математики и их приложения. 2019. № 20-1. С. 162-167. URL: https://elibrary.ru/item.asp?id=39103176 (дата обращения: 23.06.2022).

[2] Мунерман В. И. Опыт массовой обработки данных в облачных системах (на примере Windows Azure) // Системы высокой доступности. 2014. Т. 10, № 2. С. 3-8. URL: https://elibrary.ru/item.asp?id=21649609 (дата обращения: 23.06.2022); Захаров В. Н., Мунерман В. И., Самойлова Т. А. Параллельные методы вывода ассоциативных правил в технологиях in-database и in memory // CEUR Workshop Proceedings. 2018. Т. 2064. С. 219-225. URL: https://ceur-ws.org/Vol-2064/paper26.pdf (дата обращения: 23.06.2022).

Представим элементы этой таблицы как элементы гиперплоскости, а каждый столбец – их координатами. Для вычисления расстояния выберем на этой гиперплоскости метрику Таксиста, она более чем другие удовлетворяет нашим целям, поскольку не «сглаживает» разницу в нескольких конкретных координатах в угоду общей разницы. Так же определим понятие «частичного элемента» – это элемент всех координат которого мы не знаем, нам известны лишь какие-то k его координат, причем не обязательно первые k. Определим "расстояние" между частичным элементом и стандартным элементом гиперплоскости. В нашем случае, расстоянием назовем расстояние между частичным элементом, и элементом, состоящим из тех же координат целевого элемента, которые известны о частичном. Например если имеется элемент (1, 2, 3) и (?, ?, 2) то расстоянием между ними будем считать 1.

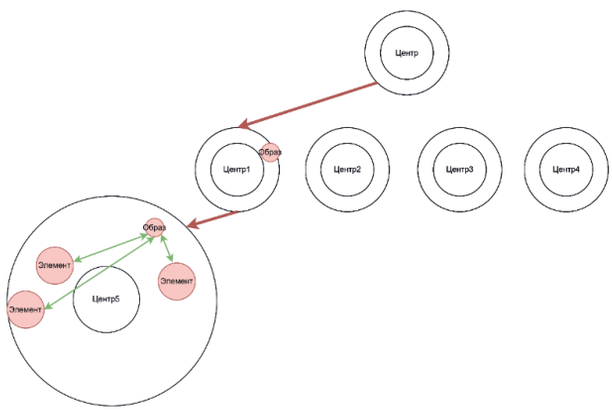
$$|0|+|0|+|3-2|=1$$

Основываясь на описанных выше определениях, начнем разбивать наше множество (на первом шаге всю таблицу) на подмножества путем использования схемы схожей с картами Кохонена. А именно выберем M элементов со случайными координатами, случайно лежащих среди элементов нашей гиперплоскости. Затем вычислим расстояние между каждым элементом таблицы и этими вершинами, при обработке каждого элемента, будем присуждать этому элементу ту вершину, которая наиболее близка к нему на данном шаге и «сдвигать» с некоторым коэффициентом выбранную вершину к обрабатываемому элементу.

Процесс можно повторять неоднократно, но это лишь немного влияет на эффективность конечной модели. После подобного передвижения назовем центрами эти вершины, определим каждый элемент таблицы к ближайшему центру, при этом для центра сохраним два значения «Максимальный радиус» и «Минимальный радиус». Первое – максимальное расстояние от центра до любого принадлежащего ему элемента. Второе – минимальное частичное расстояние от центра до принадлежащих ему элементов.

Фактически второе число – минимальная разность координат любого из принадлежащих ему элементов с центром, разумеется, если хотя бы раз оно приобрело значение нуля, сравнивать с другими элементами больше не следует.

После, будем повторять проделанную процедуру до тех пор, пока в результирующих множествах разбиения не окажется достаточно мало элементов (максимум до M дальнейшее разбиение не выгодно поскольку глубина разбиения будет в меньшей степени влиять на ширину). Отметим, что для одинаковых элементов гарантированно попадание к одним центрам, поскольку если два элемента равны между собой, они одинаково близки к другим элементам и потому два равных элемента будут отнесены к одному центру. На данном этапе имеется система, которая уже способна работать как индекс для поиска по всем столбцам таблицы. При поиске элемента достаточно проделать с ним одним операцию «примерки». А именно сравнивать его с центрами на первой «глубине» и отнести к одному из них. Затем проделать ту же процедуру на данной глубине и так далее и к все более глубоким, наиболее близким ему центрами. В конечном итоге такого сравнения поисковый образ попадет в то множества, где точно находятся/находились бы элементы эквивалентные ему. Это просто объяснить поскольку процесс попадания элементов для хранение схож, а как видно для схожим элементов получаем одни результаты. И уже среди этих элементов производить сравнение. Понять, что в таком случае скорость поиска будет склоняться с логарифмической не составляет труда – поскольку представляет из себя проход по одной ветви дерева.



Р и с. 2. Поиск

F i g. 2. Search

Необходимо отметить, что данное разбиение производится только по столбцам, по которым вообще предполагается сравнение, уникальные поля не представляют интереса и будут белым шумом при определении расстояний. Таким образом построенный индекс позволяет эффективно находить элементы по нескольким столбцам. Причем в данном случае не имеет значение порядок столбцов, что важно в поиске при классическом индексе. Так же нужно отметить, что перебалансировка такого дерева, представляется более простой, поскольку сам алгоритм вычисления каждого уровня линеен и нет необходимости для каждого столбца строить вложенное дерево.

Поиск на основе индексирования

Перейдем к вопросу о поиске образов по набору из части столбцов участвующих в таком совместном индексе. Для того чтобы по данному на вход поисковому образу определить множество для сравнения, необходимо находить ближайшие к нему центры по частичному расстоянию, а далее, не переходить к вложенным центрам, а находить описанные ваше, максимальное и минимальное частичные расстояния от выбранного центра. В случае если хранящееся для центра максимальный радиус меньше, чем расстояние от поискового образа – поиск можно прекращать, поскольку ни один из элементов базы не будет идентичен данному, что можно доказать следующим образом: Пусть R и r соответственно большой и малый радиусы, описанные выше, центра T1, а c- поисковый образ, иными словами

$$\forall a: a=(a_1,a_2,...,a_n)\in T_1:\rho(a,T_1)<R,\left|a_i-t_j\right|>r,\forall j$$

Тогда если

$$\rho(c,T_1) > R$$

и при этом T1-ближайший центр до с то

$$\forall a \in DB, a \neq c$$

Поскольку в противном случае

$$\rho(c,T_1) = \rho(a,T_1) < R$$

В случае, если максимум расстояний, хранящийся для центра, окажется большим следует сравнить минимум частичного расстояния хранящегося для центра, с минимумом частичного расстояния для данного поискового образа, в том случае, если первое окажется больше, следует перейти к сравнению со следующим центром, поскольку ни один элемент не будет эквивалентен около данного центра. Доказательство данного факта просто:
Пусть R и r соответственно большой и малый радиусы, описанные выше, центра T1 а с- поисковый образ, иными словами
Если

$$\forall \forall a : a = (a_1, a_2, ..., a_n) \in T_1 : \rho(a,T_1) < R, \left|a_i - t_j\right| > r, \forall j$$

$$\left|c_i - t_j\right| < r, \left|a_i - t_j\right| > r, \forall i$$

То само собой нет таких а, которые были бы эквивалентны с, поскольку иначе

$$Ea : \left|c_i - t_j\right| = \left|a_i - t_i\right| < r, \left|a_i - t_i\right| > r, \forall i$$

В случае же если он окажется больше, мы должны начать подобную процедуру с вложенными центрами для данного центра и так повторять далее по глубине. Причем после обхода данного центра перейти к следующему центру на этой глубине.
В итоге при прохождении каждого центра до его финальных под-центров поисковый образ будет иметь для сравнения множества элементов, которые могли бы быть потенциально близки к нему. Поскольку он близок к вложенному центру, к которому так же близки элементы базы, и заведомо не учувствует в сравнении с элементами, отсеченными по вышеописанной схеме, которые заведомо не могли быть равны этому элементу. Такой подход как видно несколько отличается от привычного индексного, поскольку обычно целью индекса является указать путь по вершинам к множеству потенциально равных элементов, тогда как для такого подхода, характерно скорее отбрасывать большое множество заведомо не равных элементов. Используя такой подход можно при помощи одной сущности определять схожие элементы, искать по полной схеме строки, а также по части из имеющихся столбцов, причем как будет показано каждый раз быстрее полного перебора, при немалой экономии дискового пространства и расходов на метаданные [16-25].

Очевидно, что при добавлении элемента в базу так же, как и при классическом индексе будет уходить дополнительное время на обслуживание добавления конкретного элемента. А именно при добавлении необходимо найти самый близкий центр, и пересчитать два его радиуса, после чего сделать аналогичную процедуру для вложенных в него центров. И в конце, в листовом множестве, желательно проверить не следует ли после очередного добавления разбить его (поскольку он может стать больше, чем M элементов).
Может показаться что процедура разбиения листового множества велика, однако нужно помнить, что элементов в листовых множествах мало и потому его разбиение не представляется трудным и в некоторых системах может осуществляться что называется «на лету».

Заключение

В настоящее время очевидным становится факт необходимости замены как можно большего числа простых в исполнении, но дающих не максимальную скорость выполнения алгоритмов на новые, способные дать не только прирост скорости, но также и новые возможности в разных областях. Разносторонность задач, выставленных перед современным компьютерным ПО, как и задач, возникающих в разных сферах деятельности, нуждаются в ведение новых принципов и подходов к разработке. В это время необходимо работать не только над фундаментальными решения, но и над обработкой частных случаев, поскольку даже частные случае при тех объемах данных, составляют заметный и ощутимый вес долях обработки данных. С другой стороны, фундаментальные решения и изменения подходов к организации обработки данных также необходимо усиливать, поскольку уже нет той возможности, при которой можно позволить мощности оборудования сглаживать углы в несовершенстве схем обработки данных. Так, видно, что нынешняя ситуация принуждает к нахождению новых не просто частичных улучшений, а принципиально новых способов обработки данных и решения разного рода программных задач с доскональным просчетом всех особенностей и частных случаев таких задач. Особенно сильно такие посылы к развитию ощущаются в сфере тех задач, которые считаются простыми заведомо, по факту своей истории и для которых, как считалось, нет нужды разработать новые методы. Такие задачи сейчас составляют особую категорию, поскольку при определенных масштабах становятся неожиданным местом полной потери производительности, примером таких операций можно считать и поиск. Как было показано выше одним из таких новых способов обработки поисковых запросов может стать значительное использование метаданных в работе алгоритмов. Подобные решения могут позволить:

- Переложение значительной части вычислительных затрат на затраты памяти;
- Освобождение от привязки к старым требования дынным для решения задач поиска;
- Решение некоторых технических проблем параллелизма;
- Оптимизацию многих видов поисковых запросов;
- Исключение линейного роста сложности;
- Обработку некоторых задач прежде, чем те будут поставлены

Из описанного выше становится понятно, что алгоритмы основанные на значительном использовании разного рода метаданных, могут стать перспективной заменой, классических, устаревающих способов решения задач. Одним из подобных решений можно считать предложенную схему исполнения поисковых запросов, которая в значительной мере оперирует метаданными базы. В данной работе была представлена схема вышеописанного алгоритма. Подобные предлагаемой схеме решения, уже включаются постепенно в жизненные циклы многих программно-аппаратных комплексов, однако необходимо усиливать участие подобных алгоритмов в разработке, что позволит существенно повысить скорость и эффективных всех этапов разработки данных, поскольку представленные проблемы находятся у истоков большинства современных алгоритмов обработки данных.

References

[1] Abdel-Basset M., Manogaran G., Abdel-Fatah L., Mirjalili S. An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. *Personal and Ubiquitous Computing*. 2018;22(5-6):1117-1132. doi: https://doi.org/10.1007/s00779-018-1132-7

[2] Chamoso P., Rivas A., Sánchez-Torres R., Rodríguez S. Social computing for image matching. *PLOS ONE.* 2018;13(5):e0197576. doi: https://doi.org/10.1371/journal.pone.0197576

[3] Das S., Grbic M., Ilic I., Jovandic I., Jovanovic A., Narasayya V.R., Radulovic M., Stikic M., Xu G., Chaudhuri S. Automatically Indexing Millions of Databases in Microsoft Azure SQL Database. In: Proceedings of the 2019 International Conference on Management of Data (SIGMOD'19). New York, NY, USA: Association for Computing Machinery; 2019. p. 666-679. doi: https://doi.org/10.1145/3299869.3314035

[4] Dodonov A., Mukhin V., Zavgorodnii V., Kornaga Ya., Zavgorodnya A., Mukhin O. Method of Parallel Information Object Search in Unified Information Spaces. *International Journal of Computer Network and Information Security.* 2021;13(4):1-13. doi: https://doi.org/10.5815/ijcnis.2021.04.01

[5] Gorokhovatskyi V.A., Gorokhovatskiy A.V., Peredrii Ye.O. Hashing of structural descriptions at building of the class image descriptor, computing of relevance and classification of the visual objects. *Telecommunications and Radio Engineering.* 2018;77(13):1159-1168. Available at: https://openarchive.nure.ua/server/api/core/bitstreams/00ab1f8f-d40e-49ee-8540-da9d745c1be4/content (accessed 23.06.2022).

[6] Graefe G. Modern B-Tree Techniques. *Foundations and Trends® in Databases*. 2011;3(4):203-402. doi: http://dx.doi.org/10.1561/1900000028

[7] Haynes D., Ray S., Manson S.M., Soni A. High performance analysis of big spatial data. In: 2015 IEEE International Conference on Big Data (Big Data). Santa Clara, CA, USA: IEEE Computer Society; 2015. p. 1953-1957. doi: https://doi.org/10.1109/BigData.2015.7363974

[8] Pan V.Y., Yu Y., Stewart C. Algebraic and Numerical Techniques for the Computation of Matrix Determinants. *Computers & Mathematics with Applications*. 1997;34(1):43-70. doi: https://doi.org/10.1016/S0898-1221(97)00097-7

[9] Kirikova A., Mironov A. Using Metadata-indexing to Improve the Efficiency of Complex Operations. In: 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia: IEEE Computer Society; 2021. p. 2124-2127. doi: https://doi.org/10.1109/ElConRus51938.2021.9396274

[10] Kirikova A., Mironov A., Munerman V. The Method of Composition Hash-functions for Optimize a Task of Searching Images in Dataset. In: 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). St. Petersburg and Moscow, Russia: IEEE Computer Society; 2020. p. 1983-1986. doi: https://doi.org/10.1109/EIConRus49466.2020.9038919

[11] Levin N.A., Munerman V.I. Models of Big Data Processing in Massively Parallel Systems. *Highly Available Systems*. 2013;9(1):035-043. Available at: https://www.elibrary.ru/item.asp?id=18928468 (accessed 23.06.2022).

[12] Lomet D. The evolution of effective B-tree: Page organization and techniques: A personal account. *ACM SIGMOD Record*. 2001;30(3):64-69. doi: https://doi.org/10.1145/603867.603878

[13] Lvovich I., Lvovich Y., Preobrazhenskiy A., Choporov O. Modeling and Optimization of Processing Large Data Arrays in Information Systems. In: 2021 International Conference on Information Technology and Nanotechnology (ITNT). Samara, Russian Federation: IEEE Computer Society; 2021. p. 1-5. doi: https://doi.org/10.1109/ITNT52450.2021.9649229

[14] Monga V., Evans B.L. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE Transactions on Image Processing.* 2006;15(11):3452-3465. doi: https://doi.org/10.1109/TIP.2006.881948

[15] Munerman V., Munerman D. Realization of Distributed Data Processing on the Basis of Container Technology. In: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). Saint Petersburg and Moscow, Russia; IEEE Computer Society; 2019. p. 1740-1744. doi: https://doi.org/10.1109/EIConRus.2019.8656766

[16] Munerman V., Munerman D., Samoilova T. The Heuristic Algorithm For Symmetric Horizontal Data Distribution. In: 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia: IEEE Computer Society; 2021. p. 2161-2165. doi: https://doi.org/10.1109/ElConRus51938.2021.9396510

[17] Alam K.S., Shishir T.A., Azharul Hasan K.M. Efficient Partitioning Algorithm for Parallel Multidimensional Matrix Operations by Linearization. In: Senjyu T., Mahalle P.N., Perumal T., Joshi A. (eds.). Information and Communication Technology for Intelligent Systems. ICTIS 2020. Smart Innovation, Systems and Technologies. Vol. 195. Singapore: Springer; 2021. p. 141-149. doi: https://doi. org/10.1007/978-981-15-7078-0_13

[18] Pushpa R. Suri, Sudesh Rani. A New Classification for Architecture of Parallel Databases. *Information Technology Journal.* 2008;7(7):983-991. doi: https://doi.org/10.3923/itj.2008.983.991

[19] Chen Y., Li K., Yang W., Xiao G., Xie X., Li T. Performance-Aware Model for Sparse Matrix-Matrix Multiplication on the Sunway TaihuLight Supercomputer. *IEEE Transactions on Parallel and Distributed Systems.* 2019;30(4):923-938. doi: https://doi.org/10.1109/TPDS.2018.2871189

[20] Sridhar R., Chandrasekaran M., Sriramya C., Page T. Optimization of heterogeneous Bin packing using adaptive genetic algorithm. *IOP Conference Series: Materials Science and Engineering*. 2017;183(1):012026. doi: https://doi.org/10.1088/1757-899X/183/1/012026

[21] Syrotkina O., Aleksieiev M., Moroz B., Matsiuk S., Shevtsova O., Kozlovskyi A. Mathematical Methods for optimizing Big Data Processing. In: 2020 10th International Conference on Advanced Computer Information Technologies (ACIT). Deggendorf, Germany: IEEE Computer Society; 2020. p. 170-176. doi: https://doi.org/10.1109/ACIT49673.2020.9208940

[22] Wajszczyk B., Gruszka I.M. Analysis of possibilities to increase the efficiency of the relative database management system using the methods of parallel processing. *Proceedings SPIE. Radioelectronic Systems Conference.* 2019;11442:1144215. doi: https://doi.org/10.1117/12.2565744

[23] Zakharov V., Kirikova A., Munerman V., Samoilova T. Architecture of Software-Hardware Complex for Searching Images in Database. In: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). Saint Petersburg and Moscow, Russia: IEEE Computer Society; 2019. p. 1735-1739. doi: https://doi.org/10.1109/EIConRus.2019.8657241

[24] Zaki M.J. Parthasarathy S., Ogihara M. Parallel Algorithms for Discovery of Association Rules. *Data Mining and Knowledge Discovery*. 1997;1:343-373. Available at: http://www.cs.rpi.edu/~zaki/PaperDir/DMKD97.pdf (accessed 23.06.2022).

[25] Zobel J., Moffat A., Sacks-Davis R. An Efficient Indexing Technique for Full Text Databases. In: Proceedings of the 18th International Conference on Very Large Data Bases (VLDB'92). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1992. p. 352-362. Available at: https://www.vldb.org/conf/1992/P353.PDF (accessed 23.06.2022).

Поступила 23.06.2022; одобрена после рецензирования 11.08.2022; принята к публикации 30.08.2022.
Submitted 23.06.2022; approved after reviewing 11.08.2022; accepted for publication 30.08.2022.

Об авторах:

Миронов Артем Игоревич, аспирант физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), **ORCID: https://orcid.org/0000-0002-6508-1943,** 615153@mail.ru

Мунерман Виктор Иосифович, доцент кафедры информатики физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), кандидат технических наук, доцент, **ORCID: https://orcid.org/0000-0002-9628-4049,** vimoon@gmail.com

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Artem I. Mironov, Postgraduate Student of the Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), **ORCID: https://orcid.org/0000-0002-6508-1943,** 615153@mail.ru

Victor I. Munerman, Associate Professor of the Chair of Computer Science, Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), Cand.Sci. (Eng.), Associate Professor, **ORCID: https://orcid.org/0000-0002-9628-4049,** vimoon@gmail.com

All authors have read and approved the final manuscript.