УДК 004.633
DOI: 10.26102/2310-6018/2025.48.1.022

Разработка программного комплекса для управления базой данных в поисковом сервисе с применением организационных систем

Д.В. Пекишев✉, А.В. Коваленко

Кубанский государственный университет, Краснодар, Российская Федерация

Резюме. В статье рассмотрена реализация системы управления базой данных, которая интересна тем, что позволяет производить быстрый поиск по статичным и неизменяемым данным, в том числе при большом объеме этих данных. Для получения результатов были с нуля разработаны программы для обработки и унификации файлов, их объединения и индексирования, а также для поиска по проиндексированным данным. Использованы методы распараллеливания, бинарного поиска, интерполяционного поиска, mmap-отображения, кластеризации, кэширования, прямой и обратной индексации, слияния, LZ-архивирования и B-деревьев. Была создана поисковая система, позволяющая выполнять тысячи поисковых запросов в секунду и работать с базами данных, размером в несколько терабайт. Актуальность исследования обусловлена необходимостью выполнения большого числа операций поиска по большим массивам данных. В связи с этим данная статья направлена на раскрытие и реализацию наиболее эффективных механизмов такого поиска. Ведущим подходом к исследованию данной проблемы является практическая реализация различных поисковых алгоритмов и их дальнейшая оптимизация для получения наиболее быстрых методов поиска. Представлены готовые алгоритмы обработки данных и дальнейших методов поиска по ним. Материалы статьи представляют собой практическую ценность для специалистов, решающих задачи, связанные с большими данными и выполняющие поисковые запросы по ним. В настоящее время такая разработка по усовершенствованию баз данных необходима из-за постоянно увеличивающегося потока цифровой информации, которую надо правильно собирать, обрабатывать, анализировать и хранить.

Ключевые слова: база данных, программный комплекс, индексация, поисковые деревья, api.

Для цитирования: Пекишев Д.В., Коваленко А.В. Разработка программного комплекса для управления базой данных в поисковом сервисе с применением организационных систем. *Моделирование, оптимизация и информационные технологии.* 2025;13(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1612 DOI: 10.26102/2310-6018/2025.48.1.022

Development of a software package for database management in a search service using organizational systems

D.V. Pekishev✉, A.V. Kovalenko

Kuban State University, Krasnodar, the Russian Federation

Abstract. The article discusses the implementation of a database management system, which is interesting because it allows for fast searches of static and unchangeable data, including large volumes of this data. To obtain the results, programs for processing and unifying files, combining and indexing them, as well as for searching by indexed data were developed from scratch. The methods of parallelization, binary search, interpolation search, mmap-mapping, clustering, caching, direct and reverse indexing, merging, LZ archiving and B-trees were used. A search engine was created that allows performing thousands of search queries per second and working with databases of several terabytes in size. The relevance of the study is due to the need to perform a large number of search operations on large arrays of data. In this regard, this article is aimed at disclosing and implementing the most effective mechanisms for such a search. The leading approach to the study of this problem is the practical

implementation of various search algorithms and their further optimization to obtain the fastest search methods. Ready-made algorithms for data processing and further methods of searching on them are presented. The materials of the article are of practical value for specialists solving problems related to big data and performing search queries on them. At present, such development for improving databases is necessary due to the constantly increasing flow of digital information that must be correctly collected, processed, analyzed and stored.

Keywords: database, software package, indexing, search trees, api.

For citation: Pekishev D.V., Kovalenko A.V. Development of a software package for database management in a search service using organizational systems. *Modeling, optimization and information technology.* 2025;13(1). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=1612 DOI: 10.26102/2310-6018/2025.48.1.022

Введение

Благодаря постоянному совершенствованию современными базами данных легко пользоваться, просто вносить в них обновления, они легко могут интегрироваться в другие системы и рассчитаны на огромную аудиторию пользователей. Для ускорения поиска по этим данным применяются самые разные методы: от разделения дисков до кэширования промежуточных точек. Большие данные характеризуются сложной структурой, в которой все данные сгруппированы по заданным параметрам, а также высокой скоростью передачи информации и большим количеством разновидностей этих данных. Поэтому так важно появление новых it-разработок, технологий, инструментов, приемов для обработки и хранения огромных и разнообразных массивов данных. В процессе работы произведен анализ следующих статей.

Так, в статье [1] обсуждается создание поисковой системы на основе алгоритма CW-Tree, который представляет собой способ индексации через построение префиксного дерева. Использование таких деревьев позволяет сильно ускорить выполнение одиночных запросов к базе данных. Алгоритм, разработанный автором статьи, помогает максимально применять возможности вычислительной системы в момент многопоточной обработки запросов. Помимо этого, уменьшает количество обращений к диску, необходимое для поиска запрашиваемых данных. Проанализировав эту статью, делаем вывод о том, что технология индексирования на основе деревьев является достаточно эффективной технологией для пространственных, многомерных, бинарных, мультимедийных и других баз данных, и такая индексация может использоваться как в постоянной, так и в оперативной памяти.

В работе [2] речь идет о текстовом поиске, который является ключевой технологией для поиска информации в интернете. В статье сравнивается поиск в текстовых системах и базах данных, акцентируется внимание на различиях, таких как типы запросов и методы оценки результатов. Авторы рассматривают алгоритмы и структуры данных, необходимые для построения высокопроизводительных индексов текста, включая методы компрессии, построения индексов и обработки запросов. Описывается практическое использование этих методов поиска и способы их оптимизации. Предлагается эффективное решение для реализации текстового поиска, объясняются базовые техники ранжирования и сжатия, а также обсуждаются возможные улучшения, такие как поддержка запросов по фразам, распределенные индексы и управление данными.

Цель и задачи исследования

В данной статье речь идет о технической разработке системы управления базой данных, предназначенной для работы с большими наборами информации, а также

уделяется внимание разработанной поисковой системе, способной выполнять расширенный поиск по данным из этой базы.

Основная задача, которая была изначально поставлена, а впоследствии выполнена, – это разработка многофункциональной, эффективно работающей базы данных, отвечающей заданным требованиям пользователей и выполняющей поисковые запросы, а также создание программного комплекса для интеграции этой базы в другие системы. Рассмотрим этапы выполненных работ для реализации поставленных задач:

– выбраны необходимые методы для полного сбора данных;
– собраны нужные данные из открытых источников;
– использован ряд сторонних API для более эффективного преобразования данных;
– для веб-скрейпинга было использовано специализированное программное обеспечение;
– исправлены появившиеся ошибки и заполнены все пропущенные значения;
– выбрана подходящая база данных – это реляционная, NoSQL, хранилище данных;
– проведена организация данных в таблицах, коллекциях и других структурах;
– реализованы обработка и анализ этих данных;
– внедрены и адаптированы для системы реляционные базы данных собственной разработки;
– определен язык программирования и использованы стандартные библиотеки;
– написана коллекция программ для обработки, объединения, сортировки, индексации данных и для выполнения поиска по данным.

Целью данного исследования является описание разработанного программного комплекса, способного выполнять быстрый поиск, использующий комбинацию специфических алгоритмов, таких, как префиксные деревья, бинарный поиск и интерполяционный поиск, по массиву заранее сохраненных и обработанных данных большого объема, которые редко подвергаются изменениям. Например, такие, как цепочки блокчейна или данные, собранные IoT, представленные в виде оригинальной NoSQL базы данных. При этом основополагающим условием стало то, что скорость поиска информации постоянна и не должна снижаться даже при очень значительном увеличении объема сохраненных данных. Для реализации поставленной выше цели были созданы авторские алгоритмы для обработки и преобразования большого объема данных, кроме того, были применены современные методы для глубокого анализа полученной информации, а также была разработана оригинальная база данных, способная увеличить скорость выполнения запросов к ней без необходимости частого изменения записей в самой базе. Такой подход важен для практического применения в том случае, когда используется относительно статичная и редко меняющаяся информация [3].

Материалы и методы

Комплекс программ был написан на языках программирования Python и PHP, при разработке использованы некоторые публичные API, такие, как Google Translate или IP Whois. Комплекс программ состоит из 4 модулей, которые отвечают за унификацию данных, индексацию данных, поиск по базе данных, а также постобработку и анализ результатов поисков. Внедренная в систему технология API позволяет интегрировать поисковую систему в сторонние приложения.

Для хранения данных используются csv-таблицы с динамическим числом столбцов и кластеризация данных с применением алгоритмов B-tree, префиксных деревьев и технологий Raid 0 и Raid 1. Для пополнения базы данных новыми данными

или удаления старых записей реализованы технологии инкрементного изменения данных, которые подразумевают невмешательство в уже сохраненные данные, и просто дополняют их новой информацией о добавленных и удаленных записях. Такой подход позволяет не нарушать структуру данных, необходимую для поиска, и при этом расширять базу без необходимости полной перестройки всего массива. Однако для снижения общего размера базы данных и большего удобства при ее использовании в сервисе целесообразно периодически, по мере приближения к максимальному доступному объему на диске, проводить полную переиндексацию.

Для сжатия данных используется алгоритм Lempel-Ziv-Welch, который сжимает информацию блоками, позволяя получать доступ к любой точке файла за константное время без необходимости расшифровки предыдущих блоков. В дополнение к этому для реализации поисковой системы были применены алгоритмы B-деревьев, бинарного и интерполяционного поиска, а также методы ускорения поиска, такие, как кэширование промежуточных точек и избыточность данных [4]. При работе с сохраненными файлами используется технология mmap, позволяющая отображать физическую память в оперативную. Кроме того, для взаимодействия с другими устройствами посредством API была применена технология long-polling. В процессе работы над проектом были изучены и после использованы методы объектно-ориентированного программирования, технологии многопоточности и распараллеливания, эвристические алгоритмы вероятностного поиска, методы прямого и обратного индексирования, а также алгоритмы глубокого поиска, позволяющие искать данные о данных. Более наглядно полный комплекс предложенных методов отображен на Рисунке 1.

Рисунок 1 – Используемые в работе методы
Figure 1 – Methods used in the project

Результаты

Рассмотрим более подробно программный комплекс поисковой системы, разработанный в ходе реализации проекта и состоящий из следующих модулей:

1) библиотека для преобразования данных в специализированный формат для дальнейшего использования;

2) библиотека для индексирования и объединения файлов, сортировки данных и создания базы данных для поиска;

3) поисковая программа, поддерживающая быстрый поиск, нечеткий поиск, поиск без учета регистра и регулярные выражения;

4) модуль интерфейса, обеспечивающий интуитивно понятное управление системой без знаний программирования;

5) API для взаимодействия с другими системами и автоматического запроса без скачивания полных баз данных;

6) модуль множественного поиска для обработки пакетных запросов и параллельных операций с несколькими дисками (RAID 1);

7) языковой модуль для перевода описаний и элементов интерфейса, поддерживает кэширование и работу с несколькими сервисами перевода.

Теперь подробно опишем структуру программного комплекса по обработке новых данных, поступающих в систему. Предполагается, что входные данные будут в форматах csv-таблиц, SQL-дампов, json-массивов или в виде простых текстовых данных. Все эти форматы необходимо в автоматическом режиме приводить к единому формату данных, который будет использоваться для дальнейшего поиска. При этом необходимо учитывать, что csv-таблицы могут иметь разные разделители и разные типы кавычек, SQL-дампы имеют различный синтаксис в зависимости от базы данных, для которой они предназначены, а json-файлы могут иметь разную разметку и хранение записей как построчно, так и с разделением одной записи на несколько строк [5].

Все эти факторы желательно учитывать при разработке программы для анализа таких данных. Количество таблиц может быть очень большим, а поэтому нецелесообразно производить независимый поиск по каждой таблице в отдельности. Также каждая таблица может содержать несколько столбцов, по которым необходимо делать поиск. Поэтому на первом этапе для каждой таблицы создается несколько копий. Все копии отличаются только порядком столбцов в них. Для каждого индексируемого столбца организуется отдельная таблица, где на первом месте стоит этот столбец, а остальные столбцы неизменны. Первый столбец также может быть преобразован для лучшего выполнения поиска. Например, если это номера телефонов, то их можно привести к международному формату, а если это ссылки, то удалить из них начало https:// или http://.

Также в каждую таблицу добавляется еще один столбец в самом конце. Он содержит индекс, по которому можно понять, из какой таблицы взята указанная строка, и восстановить заголовки этой таблицы. После формирования всех промежуточных таблиц их необходимо соединить вместе. Несмотря на то, что количество столбцов во всех таблицах может быть разное, все равно можно привести данные в первоначальный вид, просто найдя индекс из последнего столбца в записях и сопоставив предыдущие столбцы с их заголовками, что даст одну запись в формате json-словаря. Стоит заметить, что после объединения полученную таблицу нужно будет отсортировать, что будет сделать сложно, если ее размер будет очень большим, а поэтому структура должна быть более сложной, чем одна таблица. Поэтому в начале программы следует задать максимальный размер создаваемых файлов, например, на 100 Мб. Дальнейший алгоритм действий можно увидеть на Рисунке 2.

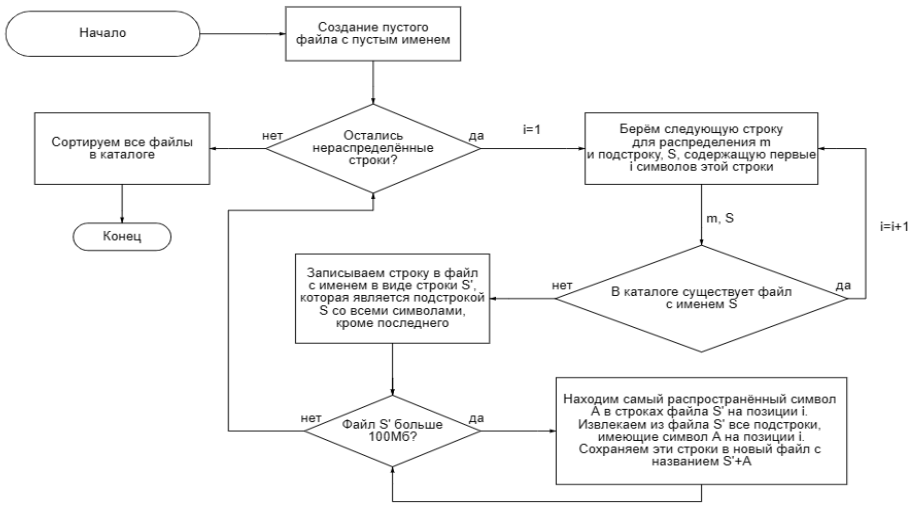

Рисунок 2 – Алгоритм сортировки и распределения данных
Figure 2 – Algorithm for sorting and distributing data

После распределения каждой таблицы происходит проверка, а именно: не превышает ли какой-либо файл указанный максимальный размер. Если находится файл, который превышает заданный размер, то строки из него разделяются на подфайлы в зависимости от (N+1)-го символа в строке, где N – длина имени файла и, соответственно, общего начала всех строк. После разделения файла на массивы строк они извлекаются в отдельные файлы, начиная с самых больших и до тех самых пор, пока размер изначального файла не станет меньше предельного размера.

После этого при необходимости итерации разделения файлов повторяются вновь уже для нового каталога файлов. Такие операции повторяются до тех пор, пока все таблицы не будут распределены по такой группе файлов. После завершения этого процесса производится сортировка всех файлов по алфавиту. При этом можно учитывать или игнорировать регистр в зависимости от того, будет ли требоваться его соблюдение при поиске [6]. Для нахождения данных в полученном массиве файлов надо последовательно выполнить следующие действия, как описано в Таблице 1.

Таблица 1 – Описание алгоритма поиска
Table 1 – Description of the search algorithm

1	Поиск файла: итерируем по строке, выбирая первые i символов, и проверяем наличие соответствующего файла, при его отсутствии откатываемся на шаг назад.
2	Поиск в файле (бинарный или интерполяционный): начальная позиция – 0, конечная – длина файла N, интерполяционный поиск быстрее при меньшем числе запросов, но бинарный эффективнее при большом объеме.
3	Обработка найденных данных: если строка меньше искомой, смещаем начальную позицию, если больше – конечную, при совпадении части строки делим поиск на два – для первого и последнего вхождения.
4	Завершение поиска: поиск заканчивается, когда диапазон становится меньше порога, при превышении лимита считываем случайную часть данных.
5	Возвращение результата: функция возвращает массив строк, проводим преобразование данных и удаляем элементы из черного списка, форматируем данные, например, преобразуем UNIX-время или телефоны в международный формат.
6	Поиск файла: итерируем по строке, выбирая первые i символов, и проверяем наличие соответствующего файла, при его отсутствии откатываемся на шаг назад.

Обсуждение

Для оценки полученных результатов было проведено сравнение с одной из популярных систем управления базами данных – SQLite. Для теста было создано 2 одинаковые базы, содержащие около 100 миллионов записей в виде строк разной длины – от 10 до 1000 символов. После создания поискового индекса по этим данным размер базы SQLite составил 18,3 Гб. Общий размер данных в разработанной системы составил 13,9 Гб.

Скорость выполнения 100 000 случайных поисковых запросов, из которых половина присутствовала в базе, а половина нет, составила 86 и 41 секунду соответственно, как на Рисунке 3. При этом в обоих случаях использовался полнотекстовый поиск, нечувствительный к регистру и порядку слов в предложении.

Сравнительный анализ показал, что комплекс программ «CombSearcher» в некоторых задачах значительно эффективнее уже существующих решений, таких как SQLite. Это подтверждает возможность использования разработанной системы для поисков по большим объемам данных в ходе решения реальных задач.

```
Тестирование базы SQLite. Количество записей в базе: 99996890
Выполнение 100 000 случайных запросов.
Время выполнения поиска: 86.26 сек.

Тестирование разработанной базы. Количество записей в базе: 99996706
Выполнение 100 000 случайных запросов.
Время выполнения поиска: 41.12 сек.
```

Рисунок 3 – Скорость выполнения поисковых запросов
Figure 3 – Search query execution speed

Заключение

В результате проведенного исследования был разработан комплекс программ «CombSearcher», позволяющий создать базу данных, нормализовать таблицы в ней и запустить поисковую систему по этим данным. Такого рода программы могут иметь широкое применение в самых разных отраслях. Разработанная база данных универсальна и может быть реализована и для любых целей, где требуется быстрый поиск по статичным и редко изменяющимся данным.

Примером таких данных могут служить цепочки исторических транзакций в blockchain, которые никогда не меняются, или данные с приборов, получаемые в IoT системах. Поисковые алгоритмы могут использоваться для обучения нейронных сетей на больших объемах данных, а также для выполнения вычислений с помощью этих нейронных систем, в ходе которых очень часто приходится искать значения весов различных нейронов и их связей.

Также подобные поисковые системы могут быть востребованы для построения деревьев связей на основе различных баз данных. В целом можно смело утверждать, что работа с большими данными очень важна и актуальна в наш век цифровых технологий и постоянных вызовов, а успешная разработка программных продуктов поможет решить проблемы, связанные с оптимизацией баз данных. Научные изыскания и сведения о разработках практических решений в сфере баз данных регулярно публикуются в научных источниках уже на протяжении нескольких десятилетий, что, несомненно, приносит неоценимую пользу для специалистов этой области [7–11].

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Шевский В.С., Шичкина Ю.А. Технология выполнения поисковых запросов к базе данных на основе метода индексации данных CW-tree. *Моделирование, оптимизация и информационные технологии*. 2021;9(1). https://doi.org/10.26102/2310-6018/2021.32.1.014
   Shevskiy V.S., Shichkina Yu.A. Technology for executing retrieval queries to a database based on the CW-tree data indexing method. *Modeling, Optimization and Information Technology*. 2021;9(1). (In Russ.). https://doi.org/10.26102/2310-6018/2021.32.1.014
2. Zobel J., Moffat A. Inverted files for text search engines. *ACM Computing Surveys*. 2006;38(2). https://doi.org/10.1145/1132956.1132959
3. Голицына О.Л., Партыка Т.Л., Попов И.И. *Основы проектирования баз данных*. Москва: Форум; 2012. 415 с.
4. Мохов В.А. Бинарная оптимизация: задачи и алгоритмы. *Известия высших учебных заведений. Северо-Кавказский регион. Технические науки.* 2022;(2):12–19. https://doi.org/10.17213/1560-3644-2022-2-12-19

Mokhov V.A. Binary optimization: problems and algorithms. *Bulletin of Higher Educational Institutions. North Caucasus Region. Technical Sciences*. 2022;(2):12–19. (In Russ.). https://doi.org/10.17213/1560-3644-2022-2-12-19

5. Демихов М.А. Методы нечеткого поиска в информационных системах. *Моделирование, оптимизация и информационные технологии*. 2015;3(2). URL: https://moit.vivt.ru/wp-content/uploads/2015/06/Demikhov_2_15_2.pdf
Demikhov M.A. The methods for fuzzy search in information systems. *Modeling, Optimization and Information Technology.* 2015;3(2). (In Russ.). URL: https://moit.vivt.ru/wp-content/uploads/2015/06/Demikhov_2_15_2.pdf
6. Hsu J.-Ch., Hsu Ch.-H., Chen S.C., Chung Ye.Ch. Correlation Aware Technique for SQL to NoSQL Transformation. In: *2014 7th International Conference on Ubi-Media Computing and Workshops, 12–14 July 2014, Ulaanbaatar, Mongolia.* IEEE; 2014. pp. 43–46. https://doi.org/10.1109/U-MEDIA.2014.27
7. Pinto Y. A Framework for Systematic Database Denormalization. *Global Journal of Computer Science and Technology.* 2009;9(4):44–52.
8. Mehmood A. ASH Search: Binary Search Optimization. *International Journal of Computer Applications.* 2019;178(15):10–17. https://doi.org/10.5120/ijca2019918788
9. Демихов М.А. Характеристики алгоритмов поиска в современных поисковых системах. *Моделирование, оптимизация и информационные технологии*. 2015;3(2). URL: https://moit.vivt.ru/wp-content/uploads/2015/06/Demikhov_2_15_1.pdf
Demikhov M.A. The characteristics of search algorithms in modern search engines. *Modeling, Optimization and Information Technology.* 2015;3(2). (In Russ.). URL: https://moit.vivt.ru/wp-content/uploads/2015/06/Demikhov_2_15_1.pdf
10. Sedgewick R., Wayne K. *Algorithms: Part I, 4th Edition.* New Jersey: Pearson Education, Inc.; 2014. 932 p.
11. Новиков Б.А., Горшкова Е.А., Графеева Н.Г. *Основы технологий баз данных*. Москва: ДМК Пресс; 2020. 582 с.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT AUTORS

Пекишев Даниил Владимирович, аспирант кафедры прикладной математики, Кубанский государственный университет, Краснодар, Российская федерация.
e-mail: AnaAble1974@gmail.com

Daniil V. Pekishev, graduate student of the Department of Applied Mathematics Kuban State University, Krasnodar, the Russian Federation.

Коваленко Анна Владимировна, доктор технических наук, доцент, заведующий кафедрой анализа данных и искусственного интеллекта, Кубанский государственный университет, Краснодар, Российская федерация.
e-mail: savanna-05@mail.ru

Anna V. Kovalenko, Doctor of Engineering Sciences, Associate Professor, Head of the Department of Data Analysis and Artificial Intelligence, Kuban State University, Krasnodar, the Russian Federation.

Статья поступила в редакцию 22.06.2024; одобрена после рецензирования 18.02.2025; принята к публикации 21.02.2025.

The article was submitted 22.06.2024; approved after reviewing 18.02.2025; accepted for publication 21.02.2025.