УДК 004.657
https://www.doi.org/10.47813/rosnio.5.2026.19.2014
EDN YUZFWL

Ресурсно-ориентированная оценка стратегий индексирования SQL-запросов в СУБД ограниченной конфигурации

Е.М. Степанова*, А.В. Письменский

Луганский государственный университет им. В. Даля, кв. Молодежный, 20а, Луганск, 91034, Россия

*E-mail: stepelen@gmail.com

Аннотация. Представлена ресурсно-ориентированная оценка стратегий индексирования SQL-запросов в реляционной СУБД ограниченной конфигурации. Цель исследования состоит в формировании и проверке матрицы выбора индексной стратегии с учётом ускорения запроса и ресурсной стоимости структуры. Методика включает анализ работ по automatic index tuning, index advisors и рекомендаций по проектированию индексов, построение предварительной матрицы, а также проверку одиночного, составного, покрывающего и фильтрованного индексирования на серии SQL-запросов. Эксперимент выполнен в Microsoft SQL Server 2022 Express на базе StackOverflow2010; оценивались длительность выполнения, логические чтения, размер индексов и изменения физических планов. Установлено, что эффективность индексирования определяется соответствием индексной структуры физическому профилю запроса и ограничениям среды. Фильтрованные индексы рациональны для устойчивых селективных подмножеств данных, составные − для запросов с фильтрацией и сортировкой, а покрывающие требуют многокритериальной оценки из-за роста размера структуры. Результаты применимы при выборе индексных стратегий для прикладных баз данных в условиях ограниченных вычислительных ресурсов.

Ключевые слова: оптимизация SQL-запросов, стратегии индексирования, реляционные СУБД, физический план выполнения, фильтрованные индексы, покрывающие индексы, ресурсные ограничения СУБД.

Resource-oriented evaluation of SQL query indexing strategies in a limited-configuration DBMS

E.M. Stepanova*, A.V. Pismenskiy

Lugansk State University named after Vladimir Dahl, 20a Molodezhnyi kv., Lugansk, 91034, Russia

*E-mail: stepelen@gmail.com

Abstract. A resource-oriented evaluation of SQL query indexing strategies in a limited-configuration relational DBMS is presented. The aim of the study is to form and verify a matrix for selecting an indexing strategy, taking into account both query acceleration and the resource cost of the index structure. The methodology includes an analysis of studies on automatic index tuning, index advisors, and index design recommendations, the construction of a preliminary matrix, and the evaluation of single-column, composite, covering, and filtered indexing on a series of SQL queries. The experiment was carried out in Microsoft SQL Server 2022 Express using the StackOverflow2010 database; query execution duration, logical reads, index size, and changes in physical execution plans were evaluated. It was found that indexing efficiency is determined by the correspondence between the index structure, the physical query profile, and the limitations of the execution environment. Filtered indexes are rational for stable selective data subsets, composite indexes are suitable for queries with filtering and sorting, while covering indexes require multi-criteria assessment due to the increased size of the structure. The results can be applied when selecting indexing strategies for application databases under limited computing resources.

Keywords: SQL query optimization, indexing strategies, relational DBMS, physical execution plan, filtered indexes, covering indexes, DBMS resource constraints.

1. Введение

Индексирование является одним из основных механизмов повышения производительности реляционных систем управления базами данных. За счёт использования индексных структур сокращается стоимость доступа к данным, уменьшается объём просматриваемых страниц и повышается эффективность выполнения SQL-запросов. Актуальность данной проблемы сохраняется как для традиционных реляционных СУБД, так и для современных систем, в которых используются различные типы индексов и разные сценарии нагрузки [1, 2]. В современных исследованиях задача индексирования всё чаще рассматривается не как изолированное ускорение отдельного запроса, а как часть настройки физической структуры базы данных с учётом профиля рабочей нагрузки, стоимости выполнения запросов, объёма хранения, ограничений среды и затрат на сопровождение индексных структур [3-5].

Несмотря на значительную изученность проблемы, практический выбор стратегии индексирования остаётся нетривиальной задачей, поскольку в литературе отсутствует единая нормативная матрица, однозначно связывающая тип SQL-нагрузки, индексную стратегию и ресурсную стоимость её применения. Работы по автоматической настройке индексов и index advisors показывают, что выбор индексной конфигурации исторически рассматривался как cost-driven и workload-aware задача: от классических инструментов подбора индексов для SQL Server и DB2 до современных методов автоматической и машинно-обучаемой настройки индексов [3-7]. Выбор индексной конфигурации зависит от структуры рабочей нагрузки, набора ограничений, способа оценки вычислительной эффективности индекса, допустимого объёма хранения и особенностей оптимизатора запросов [3, 4, 8-10].

Значимость данной проблемы усиливается распространением локальных, филиальных и распределённых информационных систем, в которых используются ограниченные по ресурсам серверные узлы. Такие системы применяются в небольших корпоративных решениях, производственных и медицинских приложениях, а также в сценариях локальной обработки данных [11-13]. Для них характерны ограничения по объёму оперативной памяти, вычислительной мощности и дисковому пространству при одновременном росте объёма хранимых и обрабатываемых данных [11].

В таких условиях индексирование приобретает характер компромисса между сокращением времени выполнения запроса и ростом затрат на хранение и сопровождение индексных структур. Избыточное количество индексов увеличивает размер базы данных,

создаёт дополнительную нагрузку на буферный кэш и дисковую подсистему, а также повышает стоимость операций вставки, обновления и удаления данных [9, 14]. Недостаточное или неудачно спроектированное индексирование, напротив, приводит к сканированию больших объёмов данных, росту количества логических чтений, выполнению дополнительных сортировок и увеличению времени выполнения SQL-запросов [1, 2, 14].

Следовательно, в ресурсно-ограниченной среде критерии выбора индексной стратегии изменяются. Если в общей постановке задачи основное внимание часто уделяется снижению времени выполнения запроса, то в ограниченной конфигурации необходимо дополнительно учитывать размер индексной структуры, влияние на буферный кэш, количество логических чтений, изменение физического плана выполнения и возможность устранения затратных операций, таких как сортировка или дополнительные обращения к основной таблице [3, 4, 9, 14].

В связи с этим возникает необходимость не только в экспериментальном сравнении отдельных индексов, но и в систематизации стратегий индексирования с позиции их применимости к различным типам SQL-нагрузки в условиях ресурсных ограничений. Такая систематизация позволяет рассматривать индекс не как изолированное средство ускорения запроса, а как элемент проектирования базы данных, влияющий на производительность, объём занимаемого пространства и устойчивость выполнения запросов [3, 4, 9].

2. Постановка задачи (Цель исследования)

В рамках статьи задача исследования состоит в переходе от общего набора рекомендаций по проектированию индексов к прикладной матрице выбора индексной стратегии, учитывающей тип SQL-нагрузки и ресурсную стоимость создаваемой индексной структуры. Такая постановка требует сопоставить базовые стратегии индексирования не только с ожидаемым ускорением запроса, но и с изменением физического плана выполнения, количеством логических чтений, размером индекса и ограничениями среды выполнения.

Целью исследования является формирование и экспериментальная проверка ресурсно-ориентированной матрицы выбора стратегий индексирования для оценки их влияния на производительность SQL-запросов с учётом ресурсных ограничений среды выполнения.

3. Методы и материалы исследования

Методика исследования включала четыре этапа: систематизацию критериев выбора индексной стратегии, формирование предварительной аналитической матрицы

применимости, экспериментальную проверку выбранных стратегий на серии диагностических SQL-запросов и уточнение матрицы с учётом ресурсных ограничений среды выполнения. На первом этапе были проанализированы работы по automatic index tuning, index advisors, cost-driven и workload-aware index selection, а также практические рекомендации по проектированию индексов в SQL Server [3, 4, 6, 7, 9, 10, 14-16]. На основе этого анализа были выделены критерии, значимые для выбора индексной стратегии: профиль рабочей нагрузки, селективность предикатов, порядок колонок в составных индексах, возможность покрытия запроса, стоимость хранения индексной структуры, влияние индекса на операции изменения данных и ожидаемое изменение физического плана выполнения.

3.1. Обоснование выбора стратегий индексирования

Современные работы в области automatic index tuning и index advisors рассматривают выбор индексной конфигурации как многофакторную задачу, зависящую от профиля нагрузки, ограничений на объём хранения, стоимости подбора индексов, динамики workload и ожидаемой пользы индексной структуры [3, 4, 8-10, 17-19]. Однако основной акцент в этих исследованиях делается на алгоритмах автоматического подбора индексов, тогда как в настоящей работе рассматривается прикладная инженерная интерпретация базовых стратегий индексирования для СУБД ограниченной конфигурации.

Для исследования были выбраны четыре стратегии индексирования: одиночное, составное, покрывающее и фильтрованное индексирование. Данный набор отражает наиболее распространённые инженерные способы изменения доступа к данным в реляционной СУБД: от минимального индекса по одному атрибуту до индексирования ограниченного подмножества строк. Выбор стратегий согласуется с современной логикой настройки индексов, где индексная конфигурация рассматривается с учётом структуры рабочей нагрузки, стоимости хранения, ограничений среды и ожидаемого влияния на план выполнения запроса [3, 4, 9, 14-16].

Одиночное индексирование рассматривается как минимальный вариант индексного доступа по одному потенциально селективному атрибуту. Такая стратегия используется как контрольный уровень между базовым вариантом без дополнительных индексов и более сложными индексными структурами. Она позволяет оценить, достаточно ли простого индекса по одному полю для улучшения запроса или структура нагрузки требует более точной настройки [14].

Составное индексирование применяется для запросов, в которых одновременно используются несколько условий отбора, сортировка или соединение таблиц. Эта стратегия позволяет проверить влияние порядка колонок в индексе, принципа левого префикса и согласованности индексной структуры с предикатами запроса и порядком сортировки [14]. В контексте исследования составные индексы важны не только как способ ускорения фильтрации, но и как средство проверки того, насколько порядок ключей индекса влияет на физический план выполнения и объём логических чтений.

Покрывающее индексирование используется для оценки ситуации, когда все необходимые для запроса данные размещаются внутри некластерного индекса, в том числе за счёт включения дополнительных неключевых столбцов. Такая стратегия позволяет уменьшить или устранить дополнительные обращения к основной таблице, однако одновременно увеличивает размер индексной структуры [15]. Поэтому она особенно важна для анализа компромисса между снижением количества чтений и ростом ресурсной стоимости индекса [9, 15].

Фильтрованное индексирование рассматривается как способ индексировать не всю таблицу, а только устойчивое подмножество строк, соответствующее предикату запроса. Эта стратегия позволяет проверить, может ли сокращение области индексирования дать выигрыш одновременно по времени выполнения, количеству логических чтений и размеру индексной структуры [16]. Для ресурсно-ограниченной среды данный вариант особенно значим, поскольку уменьшение размера индекса непосредственно связано с экономией дискового пространства и снижением нагрузки на буферный кэш [9, 16].

Тем самым каждая из выбранных стратегий задаёт отдельное направление экспериментальной проверки: простой индексный доступ, использование нескольких предикатов и порядка колонок, покрытие запроса и сокращение индексируемого пространства. Такое разделение необходимо для последующего формирования матрицы применимости, в которой стратегия индексирования сопоставляется не только с типом SQL-нагрузки, но и с ресурсной стоимостью её применения.

3.2. Формирование предварительной матрицы применимости стратегий

Предварительная матрица применимости стратегий индексирования была сформирована как авторское обобщение положений, представленных в исследованиях по автоматической и workload-aware настройке индексов, а также в практических руководствах по проектированию индексных структур [3, 4, 8-10, 14-19]. На основе систематизации этих

положений выделены критерии, определяющие применимость индексной стратегии: тип SQL-нагрузки, ожидаемое влияние на физический план выполнения, ресурсная стоимость индексной структуры и влияние на операции изменения данных. Она не рассматривается как готовый стандарт или универсальный алгоритм выбора индекса. Её назначение состоит в том, чтобы обобщить основные инженерные критерии и задать исходную схему для последующей экспериментальной проверки в условиях ресурсных ограничений.

Предварительная матрица отражает обобщённую логику выбора стратегии, не привязанную к конкретной ограниченной конфигурации СУБД: одиночные индексы связываются с простыми селективными предикатами, составные − с многокритериальной фильтрацией, соединениями и сортировкой, покрывающие − с устранением дополнительных обращений к основной таблице, фильтрованные − с обращением к устойчивому подмножеству строк [14-16]. Однако в условиях ограниченного объёма памяти, вычислительной мощности и дискового пространства эти эвристические правила требуют экспериментальной проверки и уточнения.

Таблица 1. Предварительная аналитическая матрица применимости стратегий индексирования.

Стратегия индексирования	Подходящий тип SQL-нагрузки	Ожидаемое влияние на план выполнения	Ресурсная стоимость	Влияние на операции изменения данных
Одиночное индексирование	Запросы с фильтрацией по одному достаточно селективному атрибуту; разнородные запросы, где трудно выделить устойчивую комбинацию полей	Может обеспечить индексный поиск по одному полю. При сложных условиях часто не устраняет сортировку и дополнительные обращения к основной таблице	Обычно имеет умеренный размер, но при большом количестве одиночных индексов суммарная нагрузка на хранение и буферный кэш возрастает	Каждая дополнительная индексная структура требует сопровождения при вставке, обновлении и удалении строк
Составное индексирование	Устойчивые запросы с несколькими условиями отбора, соединениями таблиц или сортировкой	Может уменьшить объём просматриваемых данных, обеспечить более точный индексный доступ и при согласовании с ORDER BY устранить отдельную сортировку	Ресурсная стоимость зависит от ширины ключа и порядка включённых колонок	Наиболее заметно влияет на операции изменения данных при обновлении ключевых колонок индекса
Покрывающее индексирование	Часто выполняемые запросы с фиксированным набором	Может устранить дополнительные обращения к основной таблице, если все	Может существенно увеличивать размер индекса за счёт включения дополнительных полей	Обслуживание индекса усложняется, особенно если часто изменяются

	возвращаемых столбцов	необходимые поля содержатся в индексе		включённые в индекс поля
Фильтрованное индексирование	Запросы к устойчивому подмножеству строк: активные записи, архивные данные, статусы, очереди обработки	Может обеспечить эффективный доступ к ограниченному набору строк при совпадении предиката запроса с условием фильтра	Обычно имеет наименьший размер, поскольку индексируется не вся таблица, а только релевантное подмножество строк	Накладные расходы возникают только при изменении строк, входящих в область действия фильтра

Данная матрица задаёт исходную исследовательскую гипотезу: разные стратегии индексирования должны проявлять наибольшую эффективность при разных типах SQL-нагрузки и различной ресурсной стоимости применения.

3.3. Обоснование выбора экспериментальных запросов

Серия экспериментальных запросов была сформирована по принципу покрытия основных физических механизмов выполнения, чувствительных к индексированию. В работах по настройке индексов значимыми признаками считаются столбцы, участвующие в фильтрации, соединениях, сортировке, группировке и возвращаемом наборе данных; при оценке вычислительной эффективности индекса также учитываются операции физического плана, связанные со сканированием, диапазонным доступом, соединением, сортировкой и группировкой данных [9, 10, 14]. Поэтому для проверки предварительной матрицы были выбраны пять диагностических запросов, соотнесённых с разными профилями SQL-нагрузки: фильтрацией и сортировкой, покрытием запроса, обращением к устойчивому подмножеству строк, соединением и агрегированием, а также проверкой порядка колонок в составном индексе.

Связь экспериментальных запросов со стратегиями индексирования показана в таблице 2.

Таблица 2. Связь экспериментальных запросов со стратегиями индексирования.

Запрос	Тип нагрузки	Проверяемый механизм	Проверяемые стратегии
Q1	Фильтрация, сортировка и ограничение результата	влияние составного индекса на способ доступа к данным и устранение сортировки	базовый вариант, составные индексы
Q2	Выборка данных пользователя	влияние одиночного, составного и покрывающего индекса на количество чтений и обращения к данным	базовый вариант, одиночный, составной, покрывающий индекс
Q3	Выборка из устойчивого подмножества строк	сравнение полного и фильтрованного индекса по длительности выполнения, логическим чтениям и размеру	базовый вариант, полный индекс, фильтрованный индекс
Q4	Соединение таблиц и агрегирование	влияние индексов на чтение связанных таблиц и физические операторы плана	базовый вариант, индексы по таблицам, участвующим в соединении
Q5	Несколько условий отбора	влияние порядка колонок в составном индексе	базовый вариант, одиночный и составные индексы

3.4. Экспериментальная среда

В качестве экспериментальной модели использована реляционная СУБД ограниченной конфигурации на примере Microsoft SQL Server 2022 Express. Данная платформа выбрана как воспроизводимая среда с явно заданными ресурсными ограничениями: экземпляр Database Engine использует не более одного сокета или четырёх ядер процессора в зависимости от того, какое ограничение достигается раньше, объём памяти для буферного кэша ограничен 1410 МБ, а максимальный размер пользовательской реляционной базы данных составляет 10 ГБ [20]. Такие ограничения позволяют моделировать класс прикладных сред, в которых объём оперативной памяти, вычислительная мощность и дисковое пространство становятся существенными факторами выбора индексной стратегии.

В качестве набора данных использована база StackOverflow2010, подготовленная для SQL Server на основе публичного дампа Stack Overflow [21]. Данный вариант базы разворачивается в пределах ограничений выбранной экспериментальной среды и содержит данные за первые годы работы Stack Overflow, включая сведения о пользователях, публикациях, комментариях, рейтингах, датах создания записей и признаках закрытия вопросов. Выбор этого набора данных обусловлен тем, что он, с одной стороны, сопоставим с предельным размером пользовательской базы данных в SQL Server Express, а с другой − сохраняет многотабличную структуру, неоднородность данных и набор типовых сущностей, характерных для реальных прикладных систем.

Отношение объёма базы данных к доступному объёму буферного кэша делает экспериментальную среду чувствительной к неудачным индексным решениям. В таких

условиях крупные или избыточные индексные структуры конкурируют с данными за память, а операции сканирования, сортировки и дополнительные обращения к основной таблице заметно отражаются на количестве логических чтений и длительности выполнения запросов. Поэтому выбранная среда подходит для проверки не только факта ускорения запросов за счёт индекса, но и ресурсной стоимости различных стратегий индексирования.

3.5. Алгоритм проведения эксперимента

Для каждого запроса Q1-Q5 формировался набор экспериментальных вариантов: базовый вариант без дополнительных экспериментальных индексов и варианты с индексами, соответствующими проверяемым стратегиям. Базовый вариант выполнялся при сохранении штатной структуры базы данных, но без специально создаваемых для эксперимента индексов. Перед переходом к очередной стратегии индексы предыдущего варианта удалялись, после чего создавался индекс текущего варианта и обновлялась статистика соответствующих таблиц.

Каждый вариант выполнялся по единой схеме: один холодный прогон после очистки буферного кэша и кэша планов (DBCC DROPCLEANBUFFERS, DBCC FREEPROCCACHE) и три тёплых прогона без повторной очистки. Холодный прогон фиксировал поведение запроса после первичного обращения к данным и повторной компиляции плана, тёплые прогоны использовались для оценки более устойчивого режима выполнения. В итоговом анализе основное внимание уделялось средним значениям тёплых прогонов, а холодные прогоны использовались как контрольные.

Для фиксации результатов использовались стандартные средства SQL Server: SET STATISTICS IO, SET STATISTICS TIME и фактические планы выполнения запросов [22–24]. Для каждого прогона фиксировались: имя запроса, вариант индексирования, режим кэша, номер итерации, длительность выполнения, количество логических чтений, CPU time, способ доступа к данным, наличие операции сортировки, наличие дополнительных обращений к основной таблице, имя созданного индекса, размер индексной структуры и имя файла сохранённого плана выполнения.

Результаты заносились в таблицу ExperimentResults. Фактические планы выполнения сохранялись отдельно и использовались для интерпретации численных показателей. Это позволило сопоставлять не только длительность выполнения и количество логических чтений, но и изменения физического плана: переход от сканирования к индексному доступу,

устранение сортировки, сокращение дополнительных обращений к данным или изменение операторов соединения и агрегирования.

Обработка результатов выполнялась в два этапа. Сначала для каждого запроса сравнивались базовый вариант и варианты индексирования. Затем результаты обобщались по стратегиям: определялись типы нагрузки, для которых стратегия дала наибольший эффект, ресурсная стоимость полученного выигрыша и выявленные ограничения. На основе этого предварительная матрица применимости стратегий индексирования уточнялась с учётом экспериментальных данных.

4. Полученные результаты

Экспериментальная проверка четырёх стратегий индексирования на пяти профилях SQL-нагрузки позволила выявить не только различия в длительности выполнения запросов, но и изменение характера использования ресурсов СУБД. Основной анализ выполнялся по тёплым прогонам, поскольку они отражают более устойчивое поведение запросов после первичного обращения к данным и прогрева кэша. При интерпретации результатов учитывались средняя длительность выполнения, количество логических чтений, способ доступа к данным, наличие сортировки, размер индексной структуры и изменение физического плана выполнения. Обобщённые результаты приведены в таблице 3.

Таблица 3. Обобщённые результаты тёплых прогонов.

Запрос	Стратегия	Среднее время, мс	Логические чтения	Способ доступа	Sort	Размер индекса, МБ
Q1	BASE, без дополнительных индексов	6099	800856	Clustered Index Scan	да	–
	COMPOSITE_FILTER	323	13301	Index Seek	да	221,13
	COMPOSITE_SORT	23	7	Index Seek	нет	221,20
Q2	BASE, без дополнительных индексов	5334	800856	Clustered Index Scan	да	–
	SINGLE (Одиночный индекс)	306	48311	Index Seek	да	50,63
	COMPOSITE_SORT	608	48323	Index Seek	нет	79,51
	COVERING (Покрывающий индекс)	552	46	Index Seek	нет	221,07
Q3	BASE, без дополнительных индексов	5968	800856	Clustered Index Scan	да	–
	FULL_DATE_INDEX (Полный индекс)	431	26404	Index Seek	нет	206,32
	FILTERED (Фильтрованный индекс)	237	614	Index Seek	нет	4,88
Q4	BASE, без дополнительных индексов	2745	169742	Clustered Index Scan	да	–
	USERS_DATE	2511	169742	Clustered Index Scan	да	11,20
	COMMENTS_USERID (Покрывающий)	830	11285	Index Scan	да	82,57

	COMBINED [a] (Комбинированный)	868	11285	Index Scan	да	93,77
Q5	BASE, без дополнительных индексов	67	7405	Clustered Index Scan	нет	–
	SINGLE (Одиночный индекс)	58	7405	Clustered Index Scan	нет	4,13
	COMPOSITE_REPUTATION_FIRST	21	309	Index Seek	нет	13,51
	COMPOSITE_LASTACCESS_FIRST	13	96	Index Seek	нет	13,51
[a] Примечание: для варианта COMBINED указан суммарный размер созданных индексных структур.						

Представленные данные показывают, что эффект индексирования проявляется неодинаково для разных профилей SQL-нагрузки. В одних случаях решающим фактором становится устранение сортировки, в других − сокращение области индексирования или уменьшение дополнительных обращений к данным. Поэтому дальнейшее обсуждение сгруппировано не по отдельным запросам, а по выявленным эффектам индексных стратегий.

1. Управление физическим порядком доступа и снижением затрат на сортировку.

Результаты по Q1 и Q5 показывают, что составной индекс в условиях ограниченного объёма памяти выступает не только средством многокритериальной фильтрации, но и инструментом управления физическим порядком доступа к данным. Такая интерпретация согласуется с общими принципами проектирования индексов, согласно которым порядок ключевых столбцов влияет на возможность индексного доступа, сокращение сканируемого диапазона и устранение дополнительных операций сортировки [10, 14].

В сценарии Q1 базовый вариант выполнялся через Clustered Index Scan, формируя 800 856 логических чтений при средней длительности выполнения 6099 мс. Создание составного индекса, ориентированного на предикаты фильтрации, изменило способ доступа на Index Seek и снизило количество логических чтений до 13 301, а длительность выполнения − до 323 мс. Однако операция сортировки в плане сохранилась, поскольку порядок ключей индекса не соответствовал полностью требуемому порядку результата.

Иной эффект был получен при использовании индекса, согласованного с секцией ORDER BY и оператором TOP. В этом случае количество логических чтений снизилось до 7, длительность выполнения − до 23 мс, а отдельная операция сортировки исчезла из плана. Это означает, что решающим фактором стала не только селективность доступа, но и соответствие порядка ключей индекса порядку получения результата.

Для ресурсно-ограниченной среды данный эффект принципиален. Оператор сортировки требует выделения памяти под промежуточный набор данных; при недостатке памяти такая операция может приводить к использованию временных структур на диске.

Поэтому устранение сортировки за счёт правильно спроектированного составного индекса снижает не только длительность выполнения, но и риск дополнительной нагрузки на память и дисковую подсистему [14, 20].

Сценарий Q5 дополнительно показал значение порядка колонок в составном индексе. При одинаковом наборе индексируемых полей вариант Reputation, LastAccessDate требовал 309 логических чтений, тогда как вариант LastAccessDate, Reputation − 96 логических чтений. Следовательно, составной индекс нельзя оценивать только по факту включения нужных столбцов. Его эффективность зависит от порядка ключей, селективности условий и фактического распределения данных [10, 14].

Таким образом, результаты Q1 и Q5 уточняют исходную матрицу применимости: в ограниченной среде составное индексирование следует рассматривать как стратегию управления не только фильтрацией, но и физическим порядком доступа, необходимостью сортировки и нагрузкой на буферный кэш.

2. Снижение пространственной стоимости за счёт фильтрованного индексирования.

Сценарий Q3 моделировал обращение к устойчивому подмножеству данных − закрытым вопросам. Такой тип нагрузки важен для прикладных систем, где запросы регулярно работают не со всей таблицей, а с логически ограниченной частью строк: архивными объектами, статусными записями, очередями обработки или закрытыми сущностями. В таких случаях фильтрованный индекс позволяет ограничить индексируемую область строками, удовлетворяющими заданному предикату, что соответствует назначению данной индексной структуры [16].

Базовый вариант требовал 800 856 логических чтений. Полный некластерный индекс по дате создания снизил количество чтений до 26 404 и уменьшил длительность выполнения до 431 мс, однако его размер составил 206,32 МБ. Фильтрованный индекс с условием ClosedDate IS NOT NULL дал более выраженный ресурсный эффект: количество логических чтений снизилось до 614, длительность выполнения − до 237 мс, а размер индексной структуры составил только 4,88 МБ.

По сравнению с полным индексом размер фильтрованного индекса оказался меньше примерно на 97,6%, а количество логических чтений снизилось более чем в 40 раз. Это показывает, что фильтрованное индексирование в данном сценарии оказалось эффективным не только по длительности выполнения, но и по стоимости хранения.

В условиях ограничения размера базы данных и объёма буферного кэша этот результат особенно значим. Полный индекс охватывает строки всей таблицы даже тогда, когда запросы

фактически обращаются к небольшому устойчивому подмножеству строк. Фильтрованный индекс уменьшает область индексирования до релевантной части данных, за счёт чего снижает объём хранимой структуры и количество страниц, вовлекаемых в выполнение запроса [9, 16, 20].

Следовательно, в ресурсно-ориентированной матрице фильтрованное индексирование должно рассматриваться как приоритетная стратегия для сценариев, где предикат отбора устойчив, подмножество строк существенно меньше полной таблицы, а экономия дискового пространства и буферного кэша имеет самостоятельное значение.

3. Границы покрывающего индексирования и изменение вычислительного профиля запроса.

Один из наиболее важных методологических результатов эксперимента связан с тем, что снижение количества логических чтений не всегда приводит к пропорциональному снижению общей длительности выполнения запроса. Это особенно заметно в сценариях Q2 и Q4, где индексирование существенно уменьшало объём чтений, но итоговая длительность зависела также от ширины индексной структуры, характера возвращаемого результата и оставшихся операторов физического плана. Такая ситуация соответствует более общей проблеме оценки вычислительной эффективности индексной конфигурации: эффект индекса не сводится к одному показателю и должен рассматриваться с учётом рабочей нагрузки, плана выполнения, стоимости хранения и ограничений среды [9, 10].

В Q2 одиночный индекс обеспечил выполнение запроса примерно за 306 мс при 48 311 логических чтениях. Покрывающий индекс снизил количество логических чтений до 46, однако средняя длительность выполнения составила около 552 мс и не стала минимальной среди индексированных вариантов. Следовательно, по метрике логических чтений покрывающий индекс оказался наиболее эффективным, но по общей длительности выполнения он не стал лучшим вариантом. Кроме того, размер покрывающего индекса составил около 221 МБ против примерно 50 МБ у одиночного индекса. Это отражает типичный компромисс покрывающего индексирования: дополнительные неключевые столбцы могут уменьшать обращения к основной таблице, но увеличивают размер индексной структуры [15].

Сходная ситуация наблюдалась в Q4. Индексирование таблицы комментариев снизило количество логических чтений со 169 742 до 11 285, то есть более чем на порядок. Однако длительность выполнения уменьшилась не пропорционально снижению чтений и составила около 830 мс. Это связано с тем, что запрос включает соединение таблиц, группировку и

сортировку по вычисленному агрегированному показателю. Даже после снижения стоимости доступа к данным часть нагрузки сохраняется на уровне обработки промежуточного результата.

Данный эффект можно интерпретировать как изменение вычислительного профиля запроса. В базовом варианте значительная часть стоимости связана с чтением большого объёма страниц. После создания индекса нагрузка на ввод-вывод уменьшается, однако относительная роль затрат на соединение, группировку, сортировку и обработку промежуточного результата возрастает. В ограниченной конфигурации СУБД, где вычислительные ресурсы также лимитированы, это препятствует линейному преобразованию выигрыша по чтениям в выигрыш по длительности выполнения [9, 10, 20].

Именно поэтому покрывающее индексирование в ресурсно-ограниченной среде требует осторожной оценки. Оно может резко уменьшить количество обращений к данным, но достигает этого ценой увеличения размера индексной структуры. Широкий индекс конкурирует с данными за буферный кэш и увеличивает объём базы данных. Поэтому его применимость должна определяться не только числом логических чтений, но и совокупностью факторов: размером индекса, длительностью выполнения, характером физического плана, наличием соединений, сортировок и агрегирования [9, 10, 14, 15].

Таким образом, результаты Q2 и Q4 уточняют матрицу выбора стратегий: покрывающий индекс не должен автоматически рассматриваться как предпочтительное решение для всех часто выполняемых запросов. В условиях ресурсных ограничений он рационален тогда, когда снижение обращений к данным компенсирует рост размера индексной структуры и не оставляет доминирующую стоимость выполнения на уровне других операторов физического плана.

Полученные результаты позволяют перейти от предварительной аналитической матрицы к ресурсно-ориентированной матрице выбора стратегий индексирования. Главное отличие уточнённой матрицы состоит в изменении акцентов: стратегия оценивается не только по способности ускорить запрос, но и по ресурсной стоимости такого ускорения. В условиях ограниченной конфигурации СУБД существенными становятся размер индексной структуры, влияние на буферный кэш, количество логических чтений, устранение затратных операций плана и устойчивость полученного эффекта для соответствующего профиля SQL-нагрузки (таблица 4).

Таблица 4. Ресурсно-ориентированная матрица выбора стратегий индексирования.

Стратегия индексиро-вания	Рациональная область применения в условиях ограниченных ресурсов	Экспериментально подтверждённый эффект	Ограничения и риски
Одиночное индексирование	Простые запросы с фильтрацией по одному достаточно селективному полю; начальный уровень оптимизации перед проектированием более сложных индексов	В Q2 сократило среднее время выполнения с 5334 до 306 мс по сравнению с базовым вариантом	Ограниченно эффективно для запросов с несколькими предикатами, сортировкой и соединениями; в Q5 не изменило количество логических чтений
Составное индексирование	Устойчивые запросы с несколькими условиями отбора, сортировкой, TOP / ORDER BY или значимым порядком доступа к данным	В Q1 индекс, согласованный с сортировкой, устранил Sort и снизил логические чтения до 7; в Q5 изменение порядка колонок снизило чтения с 309 до 96	Эффективность зависит от порядка колонок, селективности предикатов и соответствия структуре запроса; неверный порядок ключей снижает эффект
Покрывающее индексирование	Часто выполняемые запросы с фиксированным набором возвращаемых полей, где важно сократить обращения к основной таблице	В Q2 снизило количество логических чтений до 46	Увеличивает размер индексной структуры; выигрыш по чтениям не гарантирует минимального времени выполнения
Фильтрованное индексирование	Запросы к устойчивому селективному подмножеству строк: статусы, архивные записи, закрытые объекты, очереди обработки	В Q3 уменьшило размер индекса с 206,32 до 4,88 МБ и снизило логические чтения с 26404 до 614	Применимо только при устойчивом соответствии предиката запроса условию фильтра

Назначение данной матрицы состоит в предварительной инженерной оценке стратегии перед детальной настройкой запроса. Она показывает, что в ограниченной среде приоритет получают не наиболее широкие или универсальные индексные структуры, а те, которые соответствуют физическому профилю запроса и дают измеримый выигрыш без чрезмерного роста затрат на память и дисковое пространство.

Эксперимент подтвердил исходное предположение о том, что при ресурсных ограничениях меняется логика выбора индексной стратегии. Фильтрованные индексы и точно согласованные составные индексы приобретают повышенную значимость, поскольку дают выигрыш не только по длительности выполнения, но и по размеру структуры, количеству логических чтений и устранению затратных операций плана. Покрывающие индексы требуют более осторожной оценки: они могут резко уменьшать количество чтений, однако их ресурсная стоимость становится самостоятельным фактором выбора.

5. Выводы

Проведённое исследование подтвердило необходимость ресурсно-ориентированного подхода к выбору стратегий индексирования SQL-запросов в СУБД ограниченной конфигурации. В таких условиях эффективность индекса определяется не только снижением длительности выполнения запроса, но и совокупной ресурсной стоимостью: размером индексной структуры, количеством логических чтений, влиянием на буферный кэш и изменением физического плана выполнения.

Установлено, что разные стратегии индексирования имеют различные области рационального применения. Составные индексы наиболее эффективны для устойчивых шаблонов нагрузки, где важны согласование с предикатами фильтрации, порядок ключей и устранение сортировки. В эксперименте именно согласование индекса с требуемым порядком результата позволило исключить отдельную операцию сортировки из физического плана и существенно снизить объём логических чтений.

Фильтрованные индексы показали наибольшую ресурсную эффективность при обращении к устойчивым селективным подмножествам данных. Их преимущество связано с тем, что индексируется не вся таблица, а только релевантная часть строк, удовлетворяющая устойчивому предикату. За счёт этого уменьшается размер индексной структуры, снижается количество вовлекаемых страниц и уменьшается нагрузка на буферный кэш.

Покрывающие индексы требуют осторожной многокритериальной оценки. Снижение обращений к основной таблице не всегда обеспечивает пропорциональное сокращение длительности выполнения запроса и может сопровождаться существенным ростом размера индексной структуры. В условиях ограниченных ресурсов такой индекс рационален только тогда, когда выигрыш по чтениям компенсирует дополнительную нагрузку на память и дисковое пространство.

Одиночные индексы сохраняют значение как базовый уровень оптимизации для простых селективных фильтров, однако имеют ограниченную применимость для сложных профилей SQL-нагрузки, включающих несколько предикатов, сортировку, соединения или агрегирование.

Сформированная ресурсно-ориентированная матрица выбора стратегий индексирования позволяет перейти от оценки индекса как изолированного средства ускорения запроса к анализу его инженерной целесообразности в конкретной вычислительной среде. Практическая значимость предложенного подхода состоит в возможности предварительного

выбора рациональной индексной стратегии при проектировании и оптимизации прикладных баз данных, функционирующих в условиях ограниченных вычислительных ресурсов.

Перспективы дальнейшего исследования связаны с расширением матрицы на другие типы индексных структур SQL Server, например Columnstore и Memory-Optimized индексы, а также с проверкой применимости предложенного подхода в иных конфигурациях вычислительной среды.

Список литературы

1. Морозов, С.В. Сравнительный анализ типов индексов в СУБД SQL Server и PostgreSQL / С.В. Морозов, С.А. Нестеров // Системный анализ в проектировании и управлении: сборник научных трудов XXVII Международной научно-практической конференции. − 2024. − Ч. 2. − С. 485-491. − DOI: 10.18720/SPBPU/2/id24-201. − EDN: GKFRDR.
2. Золотухина, Д.Ю. Сравнительный анализ индексационных стратегий в PostgreSQL при различных сценариях нагрузки / Д.Ю. Золотухина // Программные системы и вычислительные методы. − 2025. − № 1. − С. 21-31. – DOI: 10.7256/2454-0714.2025.1.73138 − EDN: UUZPDW.
3. Wu, Y. Automatic Index Tuning: A Survey / Y. Wu, X. Zhou, Y. Zhang, G. Li // IEEE Transactions on Knowledge and Data Engineering. − 2024. − Vol. 36, No. 12. − P. 7657-7676. −DOI: 10.1109/TKDE.2024.3422006.
4. Zhou, W. Breaking It Down: An In-depth Study of Index Advisors / W. Zhou, C. Lin, X. Zhou, G. Li // Proceedings of the VLDB Endowment. − 2024. − Vol. 17, No. 10. − P. 2405-2418. − DOI: 10.14778/3675034.3675035.
5. Siddiqui, T. ML-Powered Index Tuning: An Overview of Recent Progress and Open Challenges / T. Siddiqui, W. Wu // SIGMOD Record. − 2023. − Vol. 52, No. 4. − P. 19-30. − DOI: 10.1145/3641832.3641836 .
6. Chaudhuri, S. An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server / S. Chaudhuri, V. Narasayya // Proceedings of the 23rd International Conference on Very Large Data Bases. − 1997. − P. 146-155. − URL: https://www.vldb.org/conf/1997/P146.PDF (дата обращения: 15.05.2026).
7. Valentin, G. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes / G. Valentin, M. Zuliani, D. C. Zilio [et. al.] // Proceedings of the 16th International Conference on Data Engineering. − 2000. − P. 101-110. − URL: https://www.cs.utoronto.ca/~alan/papers/icde00.pdf (дата обращения: 15.05.2026).

8. Kossmann, J. SWIRL: Selection of Workload-aware Indexes using Reinforcement Learning / J. Kossmann, A. Kastius, R. Schlosser // Proceedings of the 25th International Conference on Extending Database Technology. − 2022. − P. 155-168. − DOI: 10.48786/edbt.2022.06.
9. Wu, W. Budget-aware Index Tuning with Reinforcement Learning / W. Wu, C. Wang, T. Siddiqui [et. al] // Proceedings of the 2022 International Conference on Management of Data. – 2022. − P. 1528-1541. − DOI: 10.1145/3514221.3526128.
10. Shi, J. Learned Index Benefits: Machine Learning Based Index Performance Estimation / J. Shi, G. Cong, X.-L. Li // Proceedings of the VLDB Endowment. – 2022. − Vol. 15, No. 13. − P. 3950-3962. − DOI: 10.14778/3565838.3565848. − EDN: BUWMFH.
11. Sheikh, A. M. A Survey on Edge Computing Security Challenges / A. M. Sheikh [et. al] // Future Internet. − 2025. − Vol. 17, No. 4. – DOI: 10.3390/fi17040175.
12. Rancea, A. Edge Computing in Healthcare: Innovations, Opportunities, and Challenges / A. Rancea [и др.] // Future Internet. − 2024. − Vol. 16, No. 9. – P. 329. – DOI: 10.3390/fi16090329. – EDN: OEHDZN.
13. Nain, G. Towards Edge Computing in Intelligent Manufacturing: Past, Present and Future / G. Nain, K. K. Pattanaik, G. K. Sharma // Journal of Manufacturing Systems. − 2022. − Vol. 62. − P. 588-611. − DOI: 10.1016/j.jmsy.2022.01.010. − EDN: QBFRHW.
14. SQL Server Index Architecture and Design Guide [сайт]. – URL: https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide (дата обращения: 15.05.2026).
15. Create Indexes with Included Columns [сайт]. – URL: https://learn.microsoft.com/en-us/sql/relational-databases/indexes/create-indexes-with-included-columns (дата обращения: 15.05.2026).
16. Create Filtered Indexes [сайт]. – URL: https://learn.microsoft.com/en-us/sql/relational-databases/indexes/create-filtered-indexes (дата обращения: 15.05.2026).
17. Sharma, V. Indexer++: Workload-Aware Online Index Tuning with Transformers and Reinforcement Learning / V. Sharma, C. E. Dyreson // Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. – 2022. – P. 372-380. – DOI: 10.1145/3477314.3507691.
18. Wang, Z. Leveraging Dynamic and Heterogeneous Workload Knowledge to Boost the Performance of Index Advisors / Z. Wang, H. Liu, C. Lin [и др.] // Proceedings of the VLDB Endowment. – 2024. – Vol. 17, No. 7. – P. 1642-1654. – DOI: 10.14778/3654621.3654631. – EDN: VQIOBV.

19. Brucato, M. Wred: Workload Reduction for Scalable Index Tuning / M. Brucato, T. Siddiqui, W. Wu [и др.] // Proceedings of the ACM on Management of Data. – 2024. – Vol. 2, No. 1. – DOI: 10.1145/3639305.
20. Editions and Supported Features of SQL Server 2022 [сайт]. – URL: https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022 (дата обращения: 15.05.2026).
21. Ozar, B. How to Download the Stack Overflow Database / B. Ozar. – URL: https://www.brentozar.com/archive/2021/03/download-the-current-stack-overflow-database-for-free-2021-02/ (дата обращения: 15.05.2026).
22. SET STATISTICS IO (Transact-SQL) [сайт]. – URL: https://learn.microsoft.com/en-us/sql/t-sql/statements/set-statistics-io-transact-sql (дата обращения: 15.05.2026).
23. SET STATISTICS TIME (Transact-SQL) [сайт]. – URL: https://learn.microsoft.com/en-us/sql/t-sql/statements/set-statistics-time-transact-sql (дата обращения: 15.05.2026).
24. Display an Actual Execution Plan [сайт]. – URL: https://learn.microsoft.com/en-us/sql/relational-databases/performance/display-an-actual-execution-plan (дата обращения: 15.05.2026).