УДК 004

Организация, настройка и управление индексного пространства баз данных

Кочитов Михаил Евгеньевич
Приамурский государственный университет им. Шолом-Алейхема
студент

Аннотация
В данной статье рассматривается разработанный прототип программы, с помощью которого можно обрабатывать эффективно данные над таблицами, содержащими разные типы индексов. Также будет описана работа программы, связанная с генерацией запросов над таблицами и показаны графики проведенных экспериментов для того чтобы выявить, в какой таблице и с каким индексом обрабатываются данные максимально эффективно.
Ключевые слова: индексы, прототип, генерация запросов, обработка данных, типы индексов, база данных.

Organization, configuration and management of the database index space

Kochitov Mikhail Evgenevich
Sholom-Aleichem Priamursky State University
student

Abstract
This article discusses a developed prototype of a program with which you can efficiently process data over tables containing different types of indexes. It will also describe the work of the program associated with generating queries on tables and show graphs of the experiments in order to identify in which table and with which index the data is processed as efficiently as possible.
Keywords: indexes, prototype, query generation, data processing, index types, database.

Базы данных используются во многих организациях, веб-сайтах, приложениях и в других сферах для того, чтобы хранить в них разнообразные данные. Сами данные, хранящиеся в базах данных, логически связаны между собой и организованы в набор. Когда данные хранятся в очень большом количестве, то бывает очень трудно найти нужную запись. Записи в таблице находятся с применением метода полного сканирования таблицы, то есть происходит сканирование в таблице всех записей от самой первой до самой последней. Однако полное сканирование таблицы занимает довольно длительное время, если необходимая запись находится ближе к последней. Чтобы упростить поиск необходимых данных, то в этом случае и

используются индексы в базах данных. Применение индексов обусловлено тем, что они помогают сократить время и быстро найти необходимые данные в базах данных.

Индекс – это физическая структура данных, позволяющая осуществлять быстрый поиск нескольких данных. Настраивая индексы правильным образом можно добиться улучшения производительности запросов, которые используются для обработки реляционных данных. Поиск с использованием индексов рекомендуется использовать для таблиц, у которых очень много строк, так как они дают значительное преимущество тем, что они позволяют среди большого количества строк в таблице найти требуемые записи за очень короткий промежуток времени. Если использовать поиск без применения индексов, то будет производиться в таблице полный просмотр записей, пока не будет найдена необходимая строка, однако это занимает очень много времени, поэтому лучше применять индексированный поиск на таблицах с большим количеством разнообразных данных. Таблицы, у которых отсутствуют индексы, называются кучами и в них размещаются данные в неотсортированном виде, поэтому эти таблицы лучше использовать для хранения малого количества записей в ней.

Поэтому актуальной научно-практической задачей является разработка системы по организации, настройке и управлению индексного пространства базы данных, которая бы позволяла внедрять индексы и обрабатывать с ними данные максимально быстро и эффективно.

Цель работы – разработать систему, которая позволяет обрабатывать данные в таблицах с разными типами индексов базы данных.

Гипотеза исследования: достижение быстрой и эффективной обработки данных с помощью индексов в базе данных будет достигнуто если:

- на таблицы с данными будут добавлены индексы в нужных количествах;
- обработка данных с помощью индексов будет занимать меньше времени;
- индексы будут правильно настроены и помогут облегчить поиск данных.

Чтобы реализовать поставленную цель, то необходимо выполнить следующие задачи:

- провести анализ современных подходов к индексам в базах данных;
- выявить ключевую задачу для организации, управлении и настройке индексного пространства в базах данных;
- разработать функционал в прототипе программы для эффективной обработки данных, применяемой в базу данных над таблицами с разными типами индексов;
- провести эксперимент разработанного функционала в работу базы данных, содержащей индексируемые таблицы.

Объектом исследования является база данных с добавленными в нее таблицами, содержащими разные индексы.

Предметом исследования является разработанный прототип программы, позволяющий эффективно обрабатывать данные в индексируемых таблицах базы данных.

Практическим результатом магистерской диссертации является разработанный функционал в прототипе программы, который позволяет в базах данных обрабатывать эффективно данные в таблицах с разными индексами и среди этих таблиц определить те таблицы с индексами, которые дают возможность максимально эффективно и быстро обрабатывать в них данные.

В статье В.В. Воронина, И.В. Кочетовой и М.А. Яковлева рассматривается оптимизация производительности выполнения запросов в реляционных базах данных корпоративных информационных систем [1]. Рассматривая статью Жемеря А.В. можно увидеть особенности организации и доступа к пространственным данным в СУБД [2]. В.В. Кожушко и Д.Л. Цыганов в своей статье рассмотрели Особенности оптимизации запросов в СУБД MySQL [3]. В статье М.М. Чиркова рассматривается Оптимизация исполнения запросов к базам данных под управлением MySQL [4]. Ю.А. Григорьев и В.Г. Матюхин в своей статье рассмотрели оценку времени выполнения сложного SQL-запроса в СУБД MS SQL Server 2000 [5].

Чтобы проводить эксперимент, то необходимо разработать систему по индексированию пространства базы данных. Разработка этой системы будет происходить в интегрированной среде разработки под названием Microsoft Visual Studio 2019. Язык программирования для написания прототипа будет использоваться C#. Также будет дополнительно использоваться система управления базами данных СУБД под названием Microsoft SQL Server Management Studio 2020.

Теперь необходимо для работы разрабатываемого прототипа установить программное обеспечение Microsoft .NET Framework 4.8, так как на нем только возможно разработать необходимый функционал, чтобы провести эксперимент и получить требуемые результаты.

В базе данных прототип будет создавать четыре таблицы, которые содержат разные виды индексов. Структура таблиц одинаковая и содержит две колонки: id – идентификатор (уникальный номер) и random – случайное целое число.

Теперь рассмотрим каждую таблицу, обладающую разными типами индексов:

1. Без индекса – эта таблица не содержит никаких индексов и является неупорядоченной (кучей).

2. Некластерный – эта таблица содержит только некластерный индекс, который привязан к колонке random.

3. Кластерный – эта таблица содержит только кластерный индекс, который привязан к колонке id, содержащий первичный ключ (Primary key).

4. Оба индекса – эта таблица содержит оба индекса и кластерный на колонке id с первичным ключом (Primary key) и некластерный на колонке random.

Таблица 1 – Структура четырех таблиц

Название таблицы	Кластерный индекс (колонка ID)	Некластерный индекс (колонка RANDOM)
Без индекса	Нет	Нет
Некластерный	Нет	Есть
Кластерный	Есть	Нет
Оба индекса	Есть	Есть

Реализуемый прототип рабочей программы будет создавать генератор запросов на базу данных с этими четырьмя таблицами, в которых разные виды индексов. Прототип будет обладать возможностью с помощью четырех операций манипулирования данными (INSERT, SELECT, UPDATE, DELETE) имитировать запросы и производить их выполнение над данными в четырех таблицах с разными типами индексов.

Сам генератор запросов состоит из одного цикла, в котором задается количество итераций. Каждая итерация будет одновременно на всех четырех таблицах выполнять запрос с определенной операцией манипулирования данными (INSERT, SELECT, UPDATE, DELETE). Одновременное выполнение запросов на всех таблицах дает экономию времени, благодаря чему избавляемся от одиночного использования генератора запросов на каждую отдельно таблицу без использования параллельности.

Программа будет поддерживать работу одновременной обработки данных в этих четырех таблицах. Для этой возможности была использована многопоточность, которая позволяет на каждую таблицу подключить свой поток, который будет независимо от других потоков выполнять свою задачу и обрабатывать данные в своей таблице. Таким образом, использование многопоточности позволяет программе не зависать во время текущей работы генератора запросов, тем самым программу не придется аварийно завершать. Многопоточность в разработанном прототипе программы обязательно необходима для отображения текущего прогресса работы генератора запросов, чтобы пользователь был оповещен, на сколько процентов генератор запросов обработал данные в каждой таблице.

Результатом эксперимента в данном прототипе будет время выполнения запросов над всеми четырьмя операциями манипулирования данными в четырех таблицах. Проводимый эксперимент даст понять какая лучше таблица с индексом будет быстрее обрабатывать запросы и операции над данными самой таблицы.

Прототип тестировался на ЭВМ (электронно-вычислительной машине) со следующими характеристиками: процессор Intel Core i3-6006, оперативная

память 8 ГБ, место на жестком диске 1 ТБ. Также на этой ЭВМ была установлена и СУБД и среда по разработке самого прототипа.

Для полноценной работы прототипа необходимо запустить сначала Microsoft SQL Server с рабочей базой данных и после осуществить подключение к серверу, чтобы получить возможность выполнять имитирующие запросы над индексируемыми таблицами.

Во время разработки приложения было решено использовать библиотеку Windows Forms, которая дает возможность сделать прототип с визуальным интерфейсом, разместив в нем необходимые элементы настроек. Эти элементы необходимы для настройки прототипа так, чтобы была возможность вручную настраивать под свои нужды генератор запросов, который будет использоваться с четырьмя разными операциями над четырьмя таблицами, обладающими разными типами индексов.

Интерфейс разработанного прототипа программы представлен на рисунке ниже (рис. 2).

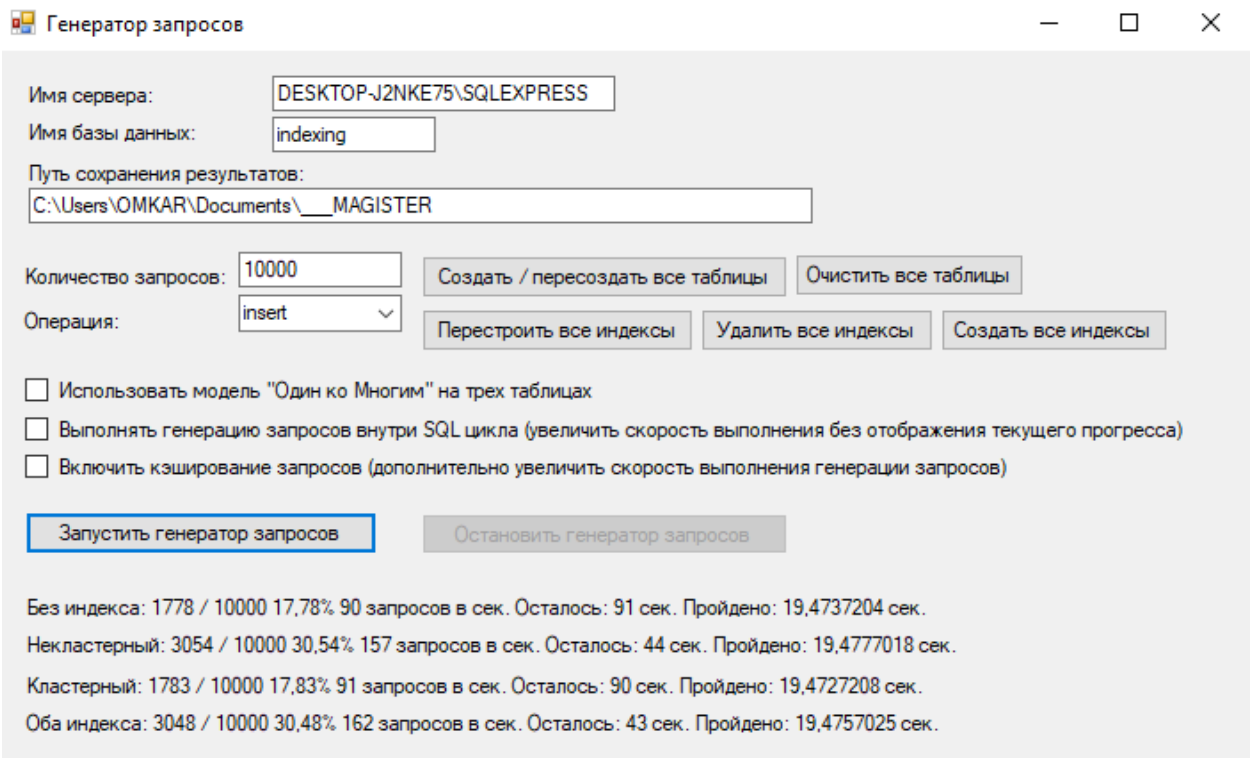

Рис. 2. Интерфейс прототипа рабочей программы с генератором запросов над таблицами с индексами

Интерфейс прототипа разработанной программы (рис. 2.1) содержит довольно много визуальных элементов для настройки программы под определенные задачи в проведении экспериментов. Сами элементы интерфейса программы разделим на группы: стартовые параметры, параметры настройки цикла, параметры настройки генератора, кнопки управления базой данных, кнопки управления процессом, результирующие надписи.

1. Стартовые параметры необходимы для подключения программы к базе данных, чтобы над ней проводить различные эксперименты. Имеются три стартовых параметра:

a. Имя сервера – этот параметр должен содержать корректное название сервера Microsoft SQL для его подключения и дальнейшей работы с ним.

b. Имя базы данных – этот параметр также должен содержать корректное название базы данных, чтобы к ней успешно подключиться и в дальнейшем с ней работать.

c. Путь сохранения результатов – в этом параметре указывается путь расположения файлов формата CSV, которые будут содержать результаты проведенных экспериментов в данном прототипе.

2. Параметры настройки цикла:

a. Количество запросов – позволяет указывать количество всех запросов, которые необходимо в цикле выполнить над всеми таблицами во время работы одного процесса.

b. Операция – этот параметр в виде выпадающего списка дает возможность выбрать для проведения эксперимента определенную операцию манипулирования данными в таблицах: insert (вставка), select (выборка), update (обновление), delete (удаление).

3. Параметры настройки генератора запросов:

a. Использовать модель «Один ко Многим» на трех таблицах – данный параметр дает прототипу провести генерацию запросов на трех таблицах со связью один ко многим. При выключенном параметре будет использоваться стандартная модель из четырех таблиц. Структура трех таблиц и их связи один ко многим будут рассмотрены ниже.

b. Выполнять транзакцию внутри SQL цикла – данная настройка позволит генератору запросов выполнять цикл внутри SQL сервера. Это существенно увеличит скорость выполнения запросов и повысит производительность над выполняемыми операциями в индексируемых таблицах, но не покажет в результирующих надписях текущий прогресс выполнения. Если параметр выключен, то цикл будет выполняться в самой оболочке программы, где за каждую итерацию будет отправляться один запрос на SQL сервер, чтобы получить от нее отклик и запись времени выполнения в секундах. Эта настройка будет полезна при обработке в цикле очень большого количества запросов, равного примерно миллиону.

c. Включить кэширование запросов – дает генератору возможность кэшировать в буфер обмена исполняемые запросы, которые позволят дополнительно увеличить производительность выполнения запросов, что даст дополнительное ускорение выполнения процесса генерации запросов. Однако при выключенном кэшировании перед каждым запросом будет очищаться буфер обмена, что замедлит процесс выполнения генерации запросов. Поэтому для обработки большого

количества запросов в цикле рекомендуется использовать кэширование.

4. Кнопки управления базой данных:

a. Создать / пересоздать все таблицы – эта кнопка при нажатии в базе данных создает четыре таблицы: без индекса, некластерный, кластерный, оба индекса. Однако при повторном нажатии существующие таблицы будут удалены и снова созданы, то есть пересозданы. В случае использования модели один ко многим в базе данных будут созданы три таблицы с необходимыми для них связями один ко многим. Структура этих таблиц с их связями будет рассмотрена ниже.

b. Очистить все таблицы – данная кнопка при нажатии удаляет содержимое всех таблиц, это нужно для того, чтобы снова проводить эксперименты операций над таблицами.

c. Перестроить все индексы – позволяет в используемых таблицах перестроить все существующие индексы, то есть их пересоздать и данные в таблицах полностью отсортировать.

d. Удалить все индексы – дает возможность удалить все индексы из таблиц, что сделает сами таблицы неупорядоченными и кучами. Эта кнопка добавлена для того, чтобы при нажатии следующей кнопки добавления индексов записать результат времени выполнения.

e. Создать все индексы – при нажатии на эту кнопку в используемых таблицах будут добавлены все необходимые индексы. Также данная кнопка в результирующие надписи записывает время выполнения одновременного создания индексов во всех четырех таблицах. Данная кнопка была предусмотрена для получения результатов времени добавления индексов в секундах на все таблицы, в которых находиться миллион строк.

5. Кнопки управления процессом. Их две:

a. Запустить генератор запросов – эта главная и важная кнопка, которая и запускает генератор запросов с настроенной операцией, количеством запросов и остальными параметрами.

b. Остановить генератор запросов – эта кнопка была реализована в случае принудительной остановки генератора запросов, если сам текущий процесс генерации запросов длиться очень долго. Аналогично этой кнопке можно закрыть саму программу.

6. Результирующие надписи показывают текущий прогресс выполнения запросов всех четырех таблиц и окончательный результат полного завершения процесса. Однако в этих надписях не отображается текущий прогресс выполнения запросов, если было включено использование генератора запросов внутри цикла SQL. На интерфейсе находится четыре строки результирующих надписей. Каждая строка связана с определенной таблицей: без индекса, некластерный, кластерный, оба индекса. Теперь разберем надпись более подробно:

a. Вид таблицы – на каждой строке располагается четыре таблицы, у которых виды: без индекса, некластерный, кластерный, оба индекса.

b. Количество выполненных запросов – здесь отображается число выполненных запросов в генераторе на данный момент.

c. Количество всех запросов – здесь показано число всех запросов, которые необходимо выполнить генератору. Это число зависит от параметра «Количество запросов».

d. Процент выполнения – показывает в процентах текущий прогресс выполнения процесса. Этот процент зависит от количества выполненных и всех запросов.

e. Скорость выполнения – отображает текущую скорость выполнения запросов в секунду. Чем выше скорость, тем быстрее выполнится процесс, а чем меньше скорость, тем медленнее будет выполняться текущий процесс генерации запросов. Число обновляется каждую секунду и определяет скорость за количество выполненных запросов в пройденную секунду.

f. Оставшееся время выполнения – показывает за счет определения текущей скорости предположительное оставшееся время выполнения текущего процесса в секундах. Число также обновляется каждую секунду.

g. Пройденное время выполнения – здесь отображается число пройденных в данный момент секунд с момента начала выполнения процесса. Во всех четырех результирующих надписях число пройденного времени в секундах одинаково. Когда в одной из четырех таблиц выполниться процесс, то пройденное время будет остановлено и покажет окончательное в секундах время выполненной генерации запросов в определенной таблице с выбранной операцией.

Генератор запросов состоит из цикла, в котором выполняется за каждую итерацию запрос с определенной операцией манипулирования данными. Далее будет показан фрагмент C# кода, в котором отображается содержимое SQL запросов на все четыре операции: insert, select, update, delete.

```
switch (type_queries[j])
{
    case "insert":
        full_name_query = "INSERT INTO dbo.rand_table" + tp.cur_table + " (random) VALUES (FLOOR(RAND()*(10000-0)+0))";
        break;
    case "select":
        full_name_query = "SELECT random FROM dbo.rand_table" + tp.cur_table + " WHERE id=" + (i + 1);
        break;
    case "update":
        full_name_query = "UPDATE dbo.rand_table" + tp.cur_table + " SET random=FLOOR(RAND()*(10000-0)+0) WHERE id=" + (i + 1);
        break;
    case "delete":
        full_name_query = "DELETE FROM dbo.rand_table" + tp.cur_table + " WHERE id=" + (i + 1);
        break;
}
SqlCommand command = new SqlCommand(full_name_query, tp.conn);
```

Рис. 3. Код формирования SQL запросов с четырьмя операциями манипулирования данными к четырем таблицам

На рисунке 3 представлен фрагмент кода, где формируются SQL запросы по четырем операциями манипулирования данными, применяемые на четыре таблицы: без индекса, некластерный, кластерный и оба индекса. Теперь рассмотрим каждую операцию более детально.

1. Операция insert предназначена для вставки новых строк в таблицу, в данном случае в столбец random будет вставляться случайное значение в диапазоне от 0 до 10000.

2. Операция select производит выборку каждой строки указанной таблицы, где указывается уникальный номер столбца id.

3. Операция update позволяет обновлять или редактировать строки таблицы, где в столбце random меняется значение на другое случайное число в таком же диапазоне 0-10000.

4. Операция delete используется для удаления строк из таблицы, где удаляется определенная строка, указываемая уникальным номером в столбце id.

В прототипе рабочей программы был рассмотрен интерфейс программы, подробно описаны все его элементы. Также был рассмотрен фрагмент кода, который представлял внутренний функционал работы генератора запросов, требуемый для полноценной работы прототипа и осуществления требуемой задачи с помощью проводимых экспериментов. Прототип приложения рабочий и готов выполнять запросы с операциями манипулирования данными над четырьмя таблицами, настроенными под разные типы индексов.

Всего 16 экспериментов будет проведено над четырьмя таблицами. Результаты первых 8 экспериментов будут представлять собой графики выполненных 10000 запросов за свое время, так как будет использован генератор запросов вне SQL цикла, что дает возможность после каждой итерации получать результат в секундах выполненного каждого запроса. На графиках по горизонтали будет отображаться количество запросов, а по вертикали – время выполнения запросов в секундах. Остальные результаты 8 других экспериментов будут представлены в виде четырех диаграмм, в которых записано окончательное время выполнения всех запросов в секунду.

Теперь проведем сначала 8 экспериментов, используя генератор запросов вне цикла SQL, и получим результаты, представленные на следующих четырех графиках (рис. 4 – 7).

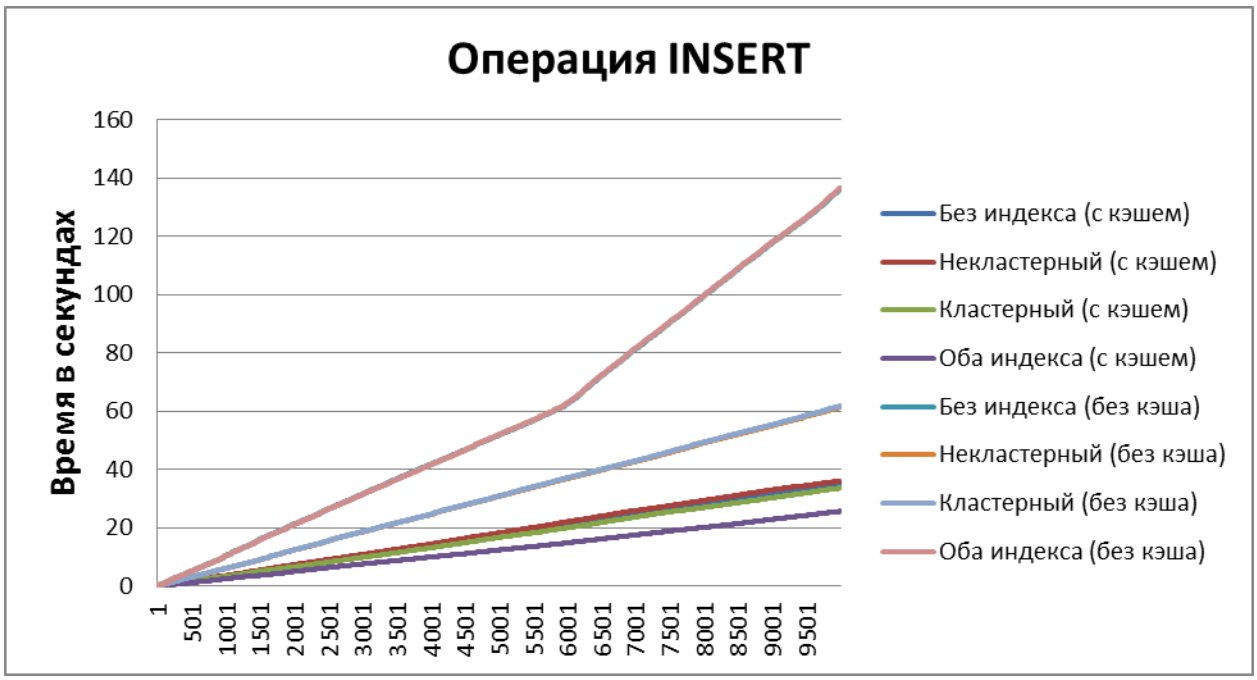

Рис. 4. Результаты выполненных 10000 запросов над четырьмя таблицами с использованием операции INSERT

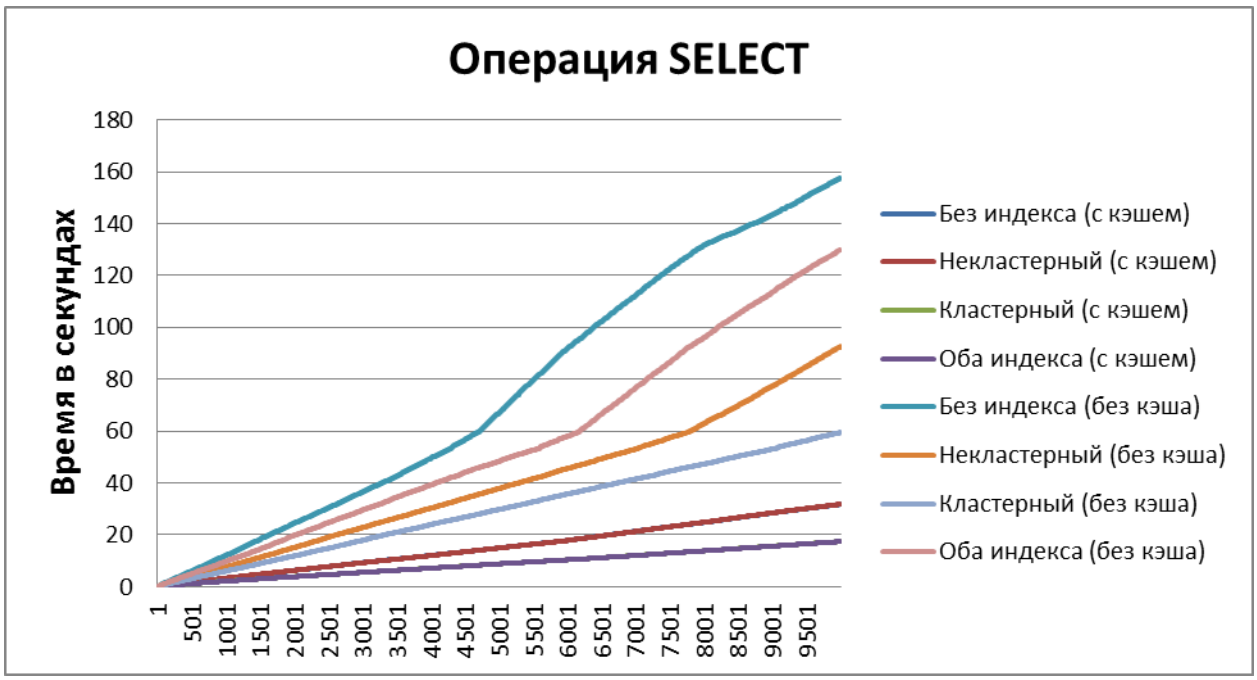

Рис. 5. Результаты выполненных 10000 запросов над четырьмя таблицами с использованием операции SELECT

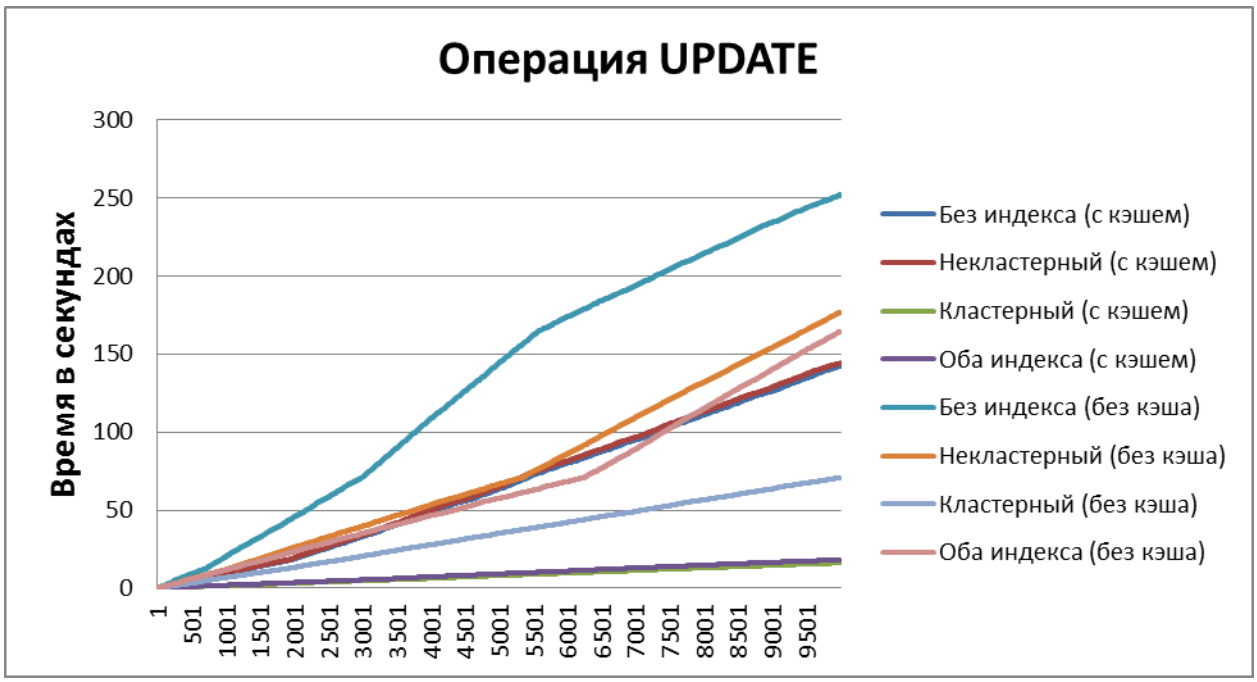

Рис. 6. Результаты выполненных 10000 запросов над четырьмя таблицами с использованием операции UPDATE

.

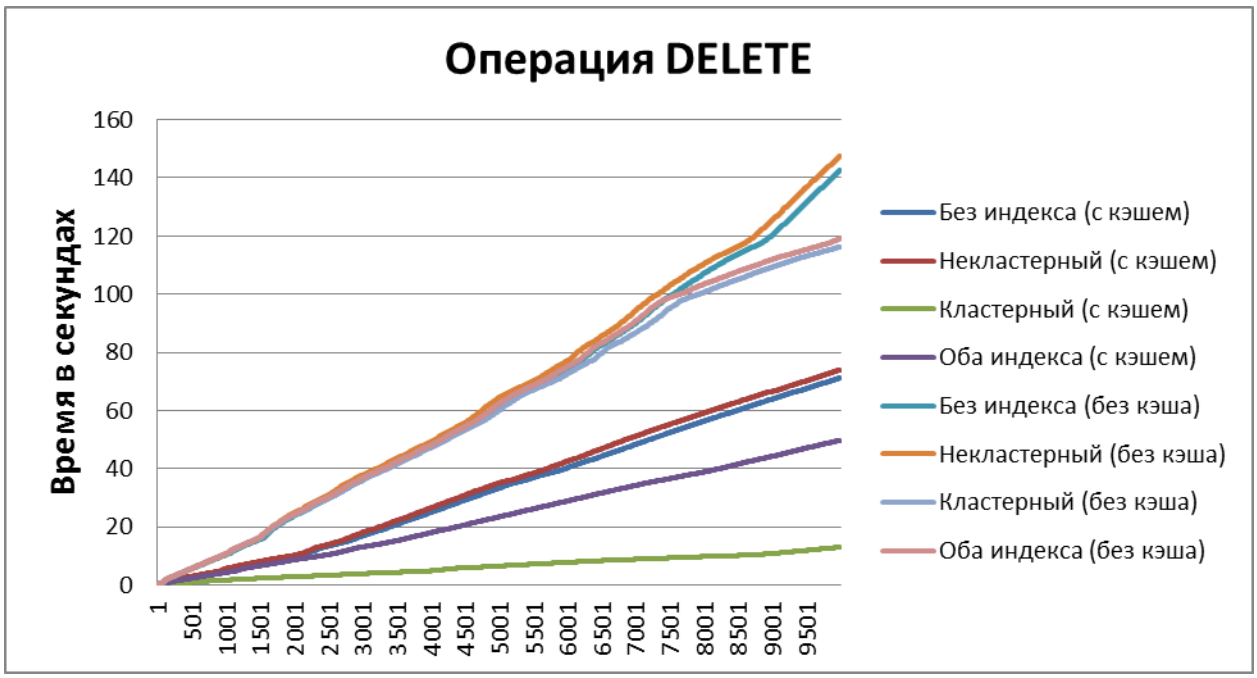

Рис. 7. Результаты выполненных 10000 запросов над четырьмя таблицами с использованием операции DELETE

Судя по результатам представленных графиков можно сделать вывод, что все операции манипулирования данными с использованием кэша запросов быстро выполняются таблицами с обоими индексами и только с кластерным индексом, то есть если в таблицах содержится кластерный индекс. Поэтому лучше обрабатывать данные в таблицах с преобладанием кластерного индекса, не очищая кэширования запросов.

Далее проведем следующие 8 экспериментов, основанные на использовании генератора запросов внутри SQL цикла и получим результаты, представленные на следующих диаграммах. Также в диаграммах будут показаны результаты и прошлых экспериментов (рис. 8 – 11).

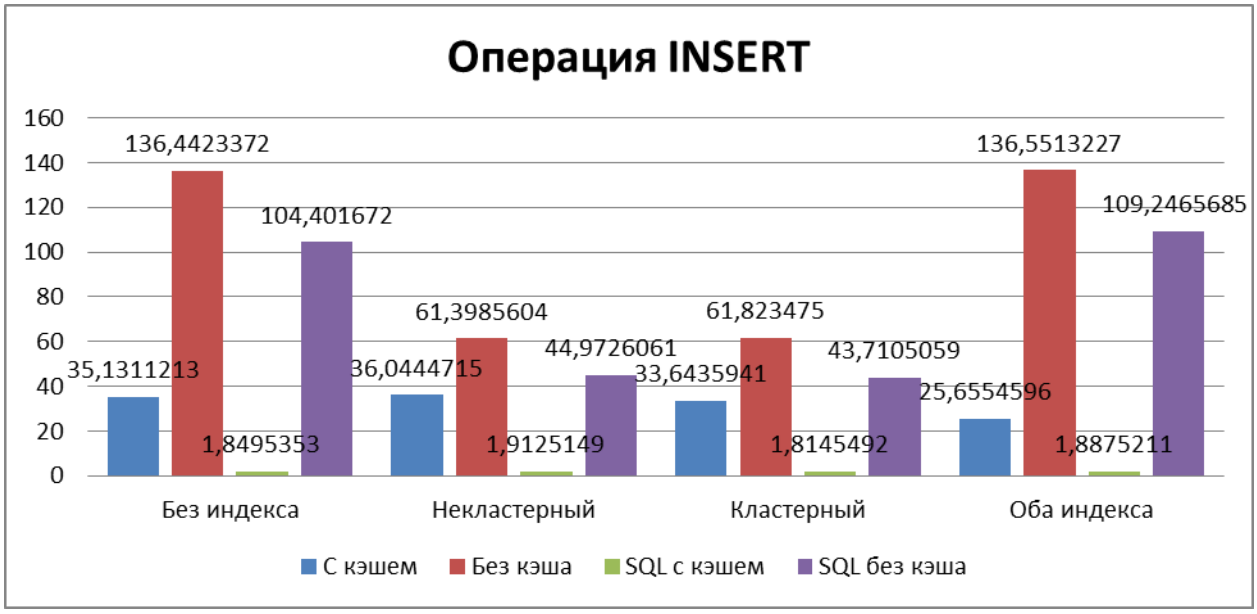

Рис. 8. Сравнение времени выполнения 10000 запросов при использовании генератора запросов вне и внутри SQL цикла над четырьмя таблицами с использованием операции INSERT

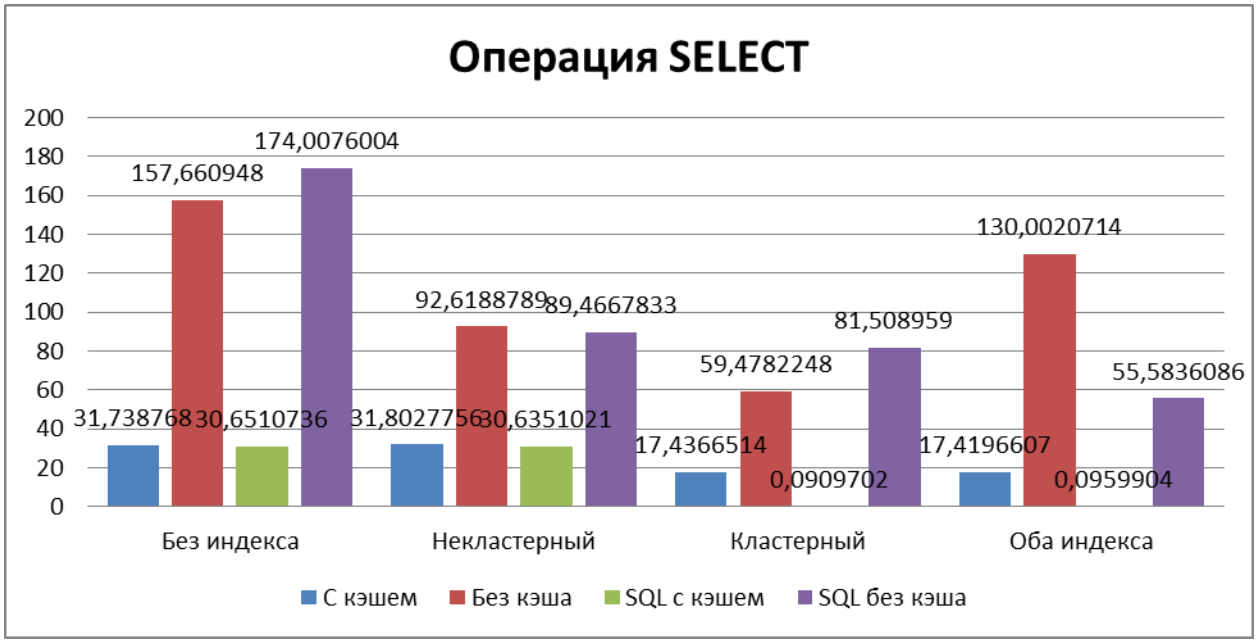

Рис. 9. Сравнение времени выполнения 10000 запросов при использовании генератора запросов вне и внутри SQL цикла над четырьмя таблицами с использованием операции SELECT

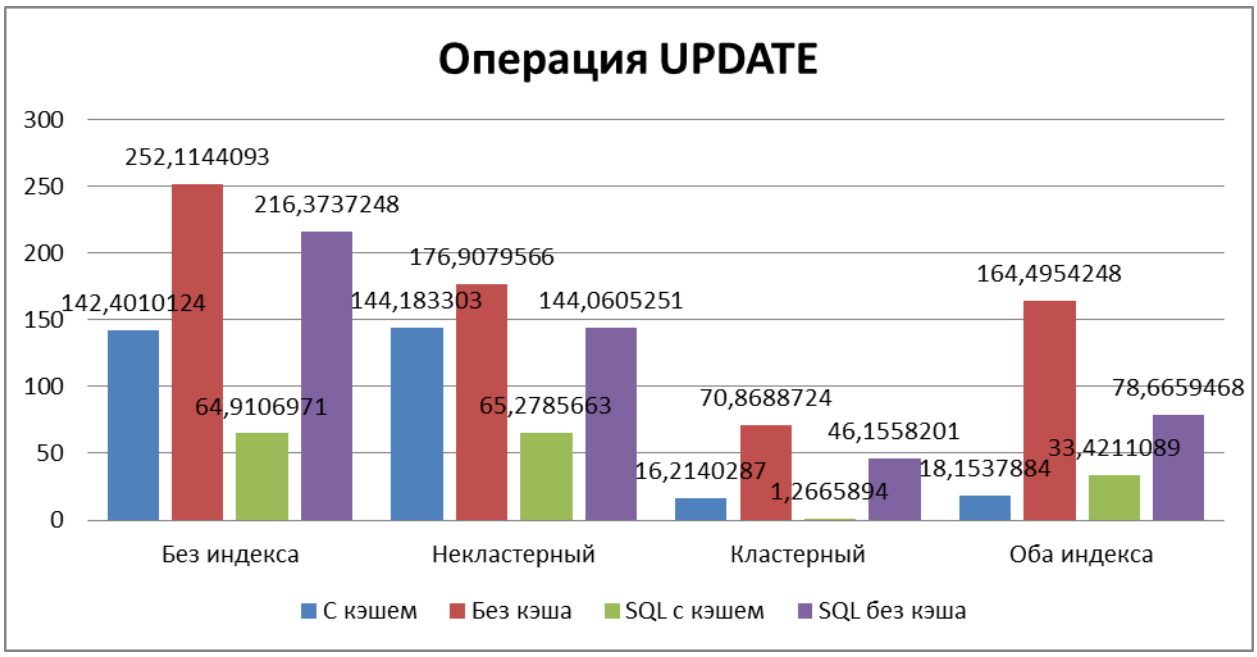

Рис. 10. Сравнение времени выполнения 10000 запросов при использовании генератора запросов вне и внутри SQL цикла над четырьмя таблицами с использованием операции UPDATE

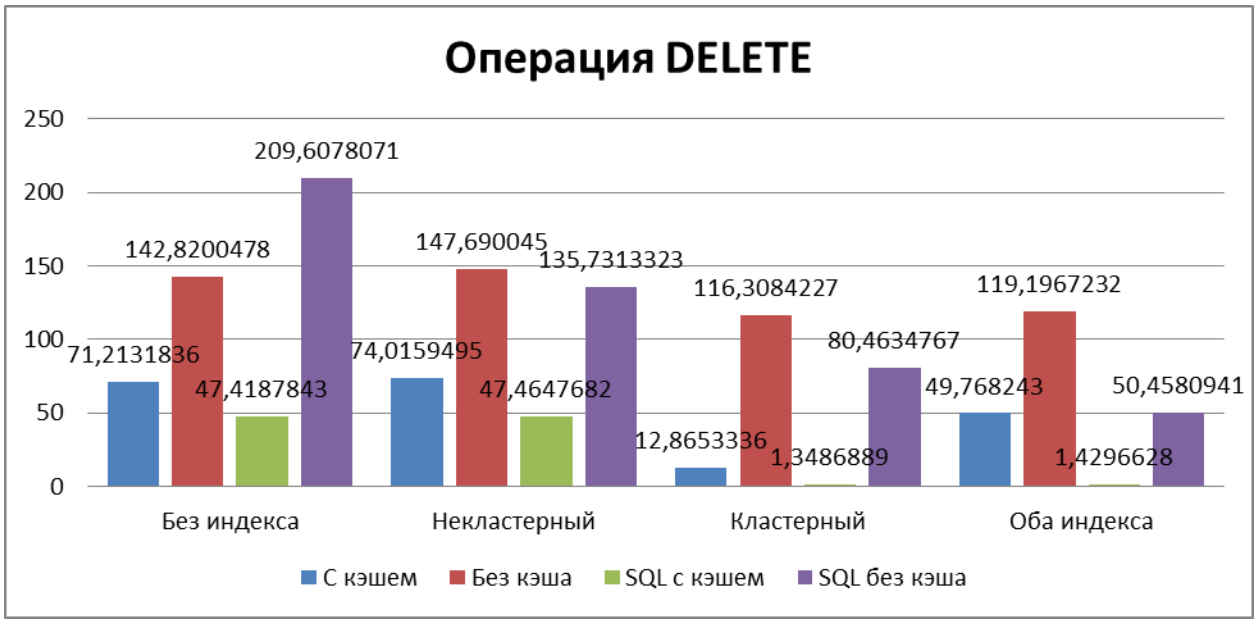

Рис. 11. Сравнение времени выполнения 10000 запросов при использовании генератора запросов вне и внутри SQL цикла над четырьмя таблицами с использованием операции DELETE

Опираясь на результаты, полученные на диаграммах можно сделать вывод, что все четыре таблицы при использовании генератора запросов внутри SQL цикла и включенным кэшированием запросов только в операции INSERT выполняют все операции за секунды при 10000 запросах. Однако таблица с только кластерным индексом также при 10000 запросах за секунды справляется со всеми операциями. Также в таблице с обоими индексами

выполняются 10000 запросов за секунды во всех операциях за исключением операции UPDATE.

Таким образом, в результате данного исследования для проведенных экспериментов был разработан прототип рабочей программы на языке программирования C#, который позволял генерировать запросы в базе данных Microsoft SQL Server в таблицах с разными типами индексов. Сама программа имела множество параметров для настройки генератора запросов, чтобы проводить множество экспериментов с целью выявления эффективной таблицы по обработке данных за короткое время, что улучшало производительность работы системы. Генератор запросов содержал два параметра, первый из которых позволял ему работать вне или внутри SQL цикла и второй давал возможность кэшировать выполняемые запросы или вовсе выполнять их без кэширования, что замедляло работу

Библиографический список

1. Воронин В.В., Кочетова И.В., Яковлев М.А. Оптимизация производительности выполнения запросов в реляционных базах данных корпоративных информационных систем // В сборнике: Информационные технологии XXI века. Материалы международной научной конференции. 2013. С. 353-359.
2. Жемеря А.В. особенности организации и доступа к пространственным данным в СУБД // Известия высших учебных заведений. Геодезия и аэрофотосъемка. 2003. № 6. С. 135-144.
3. Кожушко В.В., Цыганов Д.Л. Особенности оптимизации запросов в СУБД MySQL // Успехи современной науки и образования. 2016. Т. 5. № 7. С. 117-123.
4. Чирков М.М. Оптимизация исполнения запросов к базам данных под управлением MySQL // Вестник современных исследований. 2017. № 11-1 (14). С. 264-265.
5. Григорьев Ю.А., Матюхин В.Г. Оценка времени выполнения сложного SQL-запроса в СУБД MS SQL Server 2000 // Информатика и системы управления. 2004. № 2 (8). С. 3-13.
6. Индексы – SQL Server // Microsoft документация по SQL URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/indexes/indexes?view=sql-server-ver15 (дата обращения: 15.02.2021)
7. Transact – SQL Индексы // Professor Web URL: https://professorweb.ru/my/sql-server/2012/level3/3_5.php (дата обращения 26.02.2021)
8. Основы индексов в Microsoft SQL Server // Info Comp URL: https://info-comp.ru/programmirovanie/575-index-basics-in-ms-sql-server.html (дата обращения 05.03.2021)
9. SQL – Индексы // AndreyEX URL: https://andreyex.ru/bazy-dannyx/uchebnoe-posobie-po-sql/sql-indeksy/ (дата обращения 18.03.2021)

10. Индексы в SQL Server // IT – записки URL: https://igoldobin.wordpress.com/2016/10/23/индексы-в-sql-server/ (дата обращения 24.03.2021)
11. Типы индексов SQL Server // SQL-Ex Blog // URL: https://www.sql-ex.ru/blogs/?/Tipy_indeksov_SQL_Server.html (дата обращения 06.04.2021)
12. Индексы. Теоретические основы // SQL.RU URL: https://www.sql.ru/articles/mssql/03013101indexes.shtml (дата обращения 14.04.2021)
13. 14 вопросов об индексах в SQL Server, которые вы стеснялись задать // Хабр URL: https://habr.com/ru/post/247373/ (дата обращения 21.04.2021)
14. Индексы в среде MS SQL Server // НОУ ИНТУИТ // URL: https://intuit.ru/studies/courses/5/5/lecture/126?page=3 (дата обращения 28.04.2021)
15. T-SQL кучи, кластеризованные и некластеризованные индексы // Проект «Snake Project» Михаила Козлова URL: http://snakeproject.ru/rubric/article.php?art=tsql_indexes (дата обращения 05.05.2021)