ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента программной инженерии факультета компьютерных наук канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.Ю.Дегтярёв  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | УТВЕРЖДАЮ Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.В.Шилов  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |

**ПРОГРАММА ЛОКАЛЬНОГО ПОИСКА ДОКУМЕНТОВ ПО ИХ ИМЕНИ И СОДЕРЖИМОМУ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU. 17701729. 505900-01 81 01-1-ЛУ**

Исполнитель

студент группы 143ПИ  
\_\_\_\_\_\_\_\_\_\_\_/А.А.Смилянский/  
«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

УТВЕРЖДЁН

RU. 17701729. 505900-01 81 01-1-ЛУ

**ПРОГРАММА ЛОКАЛЬНОГО ПОИСКА ДОКУМЕНТОВ ПО ИХ ИМЕНИ И СОДЕРЖИМОМУ**

**Пояснительная записка**

**RU. 17701729. 505900-01 81 01-1**

**Листов 25**



**СОДЕРЖАНИЕ**

[**1. Введение 4**](#_Toc480249547)

[**1.1. Наименование программы 4**](#_Toc480249548)

[**1.2. Документы, на основании которых ведётся разработка 4**](#_Toc480249549)

[**2. Назначение и область применения 5**](#_Toc480249550)

[**2.1. Назначение программы 5**](#_Toc480249551)

[**2.1.1. Функциональное назначение 5**](#_Toc480249552)

[**2.1.2. Эксплуатационное назначение 5**](#_Toc480249553)

[**2.2. Краткая характеристика области применения 5**](#_Toc480249554)

[**3. Технические характеристики 6**](#_Toc480249555)

[**3.1. Постановка задачи на разработку программы 6**](#_Toc480249556)

[**3.2. Описание алгоритма и функционирования программы 6**](#_Toc480249557)

[**3.2.1. Описание алгоритма и функционирования программы для поиска файлов в указанных директориях 6**](#_Toc480249558)

[**3.2.2. Описание алгоритма индексирования 8**](#_Toc480249559)

[**3.2.3. Описание алгоритма работы программы 10**](#_Toc480249560)

[**3.2.4. Алгоритмы работы обработчиков и вызываемых функций 12**](#_Toc480249561)

[**3.3. Описание и обоснование выбора метода организации входных и выходных данных 12**](#_Toc480249562)

[**3.3.1. Описание метода организации входных и выходных данных 12**](#_Toc480249563)

[**3.3.2. Обоснования выбора метода организации входных и выходных данных 12**](#_Toc480249564)

[**3.4. Описание и обоснование выбора составных технических и программных продуктов 13**](#_Toc480249565)

[**3.4.1. Состав технических и программных средств 13**](#_Toc480249566)

[**3.4.2. Обоснование выбора технических и программных средств 14**](#_Toc480249567)

[**4. Ожидаемые технико-экономические показатели 15**](#_Toc480249568)

[**4.1. Предполагаемая потребность 15**](#_Toc480249569)

[**4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами 15**](#_Toc480249570)

[**5. Список использованных источников 20**](#_Toc480249571)

[**Приложение 1. Терминология 21**](#_Toc480249572)

[**Приложение 2. Основные алгоритмы 22**](#_Toc480249573)

[**Приложение 3. Основные обработчики 23**](#_Toc480249574)

[**Приложение 4. Обоснование принятых решений 24**](#_Toc480249575)

[**Лист регистрации изменений 25**](#_Toc480249576)

1. **ВВЕДЕНИЕ**
   1. **Наименование программы**

Наименование программы – «Программа локального поиска документов по их имени и содержимому». Наименование на английском языке – «The Program for Local Search of Files Based on Their Names and Contents».

* 1. **Документы, на основании которых ведётся разработка**

Программа выполнена в рамках темы курсовой работы «Программа локального поиска документов по их имени и содержимому» (факультет компьютерных наук, департамент «программная инженерия»), в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

Документы:

1) Приказ Национального исследовательского университета "Высшая школа экономики" № 2.3-02/0812-01 от 08.12.2016.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Назначение программы**
      1. **Функциональное назначение**

Функциональным назначением данного приложения является просмотр существующих на устройстве файлов с целью определения из них тех, что содержат хотя бы одно вхождение интересующего слова. Вхождения могут встречаться как непосредственно в содержимом, так и в названии, расширении. Зачастую, ставится задача нахождения каталогов.

* + 1. **Эксплуатационное назначение**

Решение задачи поиска файлов является востребованным на сегодняшний день, так как затрагивает множество областей применения и используется в большинстве программных средств, в задачах которых стоит работа с файловой системой. Задача поиска по файлам находит применения при решении таких задач, как:

1. Работа системы веб поиска, в том числе индексирование и поиск среди веб-ресурсов, веб-страниц, файлов на удалённых серверах и др.
2. Делопроизводство, поиск документов по различным проектам, поиск по датам создания
3. Проверка качества работы запоминающих устройств по времени поиска файлов, по полноте найденной информации  
   1. **Краткая характеристика области применения**

Задача стоит в просмотре существующих на устройстве файлов с целью определения из них тех, что содержат хотя бы одно вхождение интересующего слова. Слова могут встречаться как непосредственно в содержимом, так и в их названии, расширении. Зачастую, ставится задача нахождения каталогов.

Если переходить к повседневной терминологии, то данный проект позволяет найти на устройстве файлы в которых хотя бы один раз встречается заданное слово. Данная проблема затрагивается в работах, связанных с внесением изменений в файлы, что является одной из задач электронно-вычислительных машин.

Ряд областей применения:

1. Делопроизводство
2. Программирование
3. Дизайн
4. Монтаж видеофайлов

Один из наиболее известных и часто используемых примеров применения данной задачи – поисковые системы, позволяющий находить информацию среди веб-страниц.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Разрабатываемое решение должно:

1. Выполнять поиск файлов в указанной (-ых) директории (-ях);
2. Создавать индексы с указанными характеристиками и по указанным директориям для быстрого поиска;
3. Выполнять сохранение созданных индексов по завершению работы программы;
4. Выполнять загрузку созданных ранее индексов при загрузке программы;  
   1. **Описание алгоритма и функционирования программы**
      1. **Описание алгоритма и функционирования программы для поиска файлов в указанных директориях**

Алгоритм порядка работы с программой – поиска файлов в указанных директориях приведён ниже (рис.1.):

C:\Users\Александр\Downloads\Alg_1.png

*Рис.1.* Алгоритм поиска файлов с использованием индекса (обобщённый)

Сложность и разбор алгоритма описаны в [Приложение 2. Основные алгоритмы].

* + 1. **Описание алгоритма индексирования**

Алгоритм обработки запросов на индексирование представлен ниже (Рис.2.):C:\Users\Александр\Downloads\Alg_2.png

*Рис. 2.* Основной поток выполнения запроса на индексирование.

Алгоритм обработчика индексирования содержимого представлен ниже (Рис. 3):

C:\Users\Александр\Downloads\Alg_3.png

*Рис. 3.* Алгоритм обработчика индексирования содержимого.

Сложность и объяснение размера партии представлены в [Приложение 2. Основные алгоритмы].

Создание буфера обусловлено тем, что включения находятся часто, в связи с этим записи в индекс должны производится так же часто, но если индекс – это база данных, то на каждую запись требуется открывать транзакцию, что влечёт затрату больших ресурсов. Однако, сохранение включений во временную структуру данных (List), позволяет снизить число обращений к индексу, а значит уменьшить количество транзакций в случае структуры индекса – базы данных.

* + 1. **Описание алгоритма работы программы**

Алгоритм работы программы представлен ниже (Рис. 4):

C:\Users\Александр\Downloads\Alg_4.png

*Рис. 4.* Алгоритм работы программы

Индексы загружаются в отдельном потоке, при запуске программы. Сохранение их происходит при закрытии программы. Обоснование выбора количества потоков представлено в [Приложение 2. Основные алгоритмы].

* + 1. **Алгоритмы работы обработчиков и вызываемых функций**

Обработчики (Executors) являются частью пакета “java.util.concurrent” и предоставляют возможность одновременного вызова и контроля ряда процессов. В программе применяется реализация ThreadPoolExecutor данного интерфейса, предоставляющая методы для обработки одновременного выполнения N-ого количества потоков. Данный механизм используется в местах, требующих многопоточности, в частности: выполнение заданий на поиск и индексирование, индексирование содержимого файлов. Рассматривался вариант ограничения доступа к хранилищу индекса [Приложение 3. Структуры], однако в процессе замены семафоров [Приложение 1. Терминология] потребность в этом отпала.

Общий алгоритм выполнения – ожидание запуска, получение задания, создание потоков для выполнения, назначение потока на выполнения задания, возвращение к пункту получение задания, по завершению всех заданий – закрытие потоков выполнения.

Вызываемые функции (Callable) являются частью пакета “java.util.concurrent” и служат для передачи задания (вызываемый метод, возвращающий результат) потокам Executor. В работе применялись дополненный JavaFx вариант – Task.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Выходные данные (данные индекса) формируются во время выполнения программы и уникальны для всех индексируемых директорий. Они не могут быть применены на другом устройстве, т.к. структура каталогов может полностью не совпадать с той, на которой был построен индекс.

Индекс формируется по данным параметрам и на основе указанных директориях. Вручную, пользователь может изменять список директорий индексирование.

Входные данные для первого запуска программы не предоставляются, для последующие – сформированные индексы, расположенные в каталоге “indices” директории с исполняемым файлом.

Файлы делятся на два типа:

* \*.ser – сериализованные данные о параметрах индекса и проиндексированных директориях
* \*.db – данные хранилища индекса [Приложение 1. Терминология] о словах, директориях и включениях.  
  + 1. **Обоснования выбора метода организации входных и выходных данных**

Механизм сохранения индекса и его хранилища был выбран на основе проведённых наблюдений за поведением ряда алгоритмов во время индексирование и загрузки/сохранения. Сериализация позволяет сохранять данные объекта в файл и загрузить его из него, однако, не предназначена для постоянного обращения на запись. Индексирование подразумевает сохранение большого количества информации о файлах за отдельные промежутки времени. Даже алгоритм буферизации (см. рис.3, пункт 3.2.2.) не полностью решает проблему (существенные затраты ресурсов процессора и оперативной памяти), т.к. требуют большого числа буферизованных включений для избегания постоянного процесса сериализации. К тому же, встроенный в Java механизм сериализации обладает рядом существенных недостатков, самый главный из которых – большое время чтения/записи.

По причине невозможности хранения большого индекса в оперативной памяти (размер индекса существенно растёт с увеличением размера индексируемых данных) была выбрана модель базы данных. В качестве драйвера базы данных использовалось JDBC, однако, было решено использовать ORM модель с использованием фреймворка JPA - Hibernate. Это означает, что ряд объектов, сохраняющихся в базе данных (БД) в последствии можно изменять и изменения будут сохраняться в БД. Для сохранения изменений, используются сессии hibernate и транзакции jpa.

Задачи БД хранилища – сохранение буфера включений в базу данных, проверка на существование записей, выдача результата по заданному слову.

Для подключения к базе данных, требуются параметры индекса и его название, поэтому сохраняются посредством сериализации походу выполнения программы. Это позволяет сократить количество кода и не потерять данные индекса по завершению работы программы.

* 1. **Описание и обоснование выбора составных технических и программных продуктов**
     1. **Состав технических и программных средств**

Требуется устройство, с установленной Java SE 8. Рассчитано для работы на системе Microsoft Windows, работа на других системах может отличаться от предъявленных критериев. В список отличий может входить:

1. Отображение элементов интерфейса
2. Порядок выполнения потоков программы
3. Порядок и настройки индексирования

Для работы минимально необходимо:

* RAM: 512 МБ
* дисковое пространство: 256 МБ (JRE)
* компьютерная мышь
* Процессор: минимальное требование - Pentium 2 266 МГц
* Разрешение – не менее 430x290 (минимальное разрешение главного окна программы)  
  + 1. **Обоснование выбора технических и программных средств**

Описание обоснование выбора хранилища индекса приведено в пункте 3.3.2. Обоснование выбора метода организации входных и выходных данных. Данная программа при работе обладает следующими зависимостями от библиотек:

1. Javax.ejb
2. Org.hibernate
3. Com.h2database
4. Javax
5. Javafx

Данные библиотеки включены в запускаемый файл \*.jar в процессе компиляции по средствам системы сборки.

Выбор h2 базы данных был обоснован распространённостью данного фреймворка, позволяющего создавать встроенные (embedded) базы данных, в связи с чем обладает большим количеством примеров и хорошо документирован.

Для работы необходима одна из следующих операционных систем (условие работы Java 8):

* Windows 10 (8u51 или более поздняя)
* Windows 8.x (настольная версия)
* Windows 7 с пакетом обновления 1 (SP1)
* Windows Vista SP2
* Windows Server 2008 R2 с пакетом обновления 1 (SP1) (64-разрядная версия)
* Windows Server 2012 и 2012 R2 (64-разрядная версия)

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Предполагаемая потребность**

Решение задачи поиска файлов является востребованным на сегодняшний день, так как затрагивает множество областей применения и используется в большинстве программных средств, в задачах которых стоит работа с файловой системой. Программа может быть использована офисными работниками, студентами, программистами, дизайнерам для поиска файлов ранее созданных на компьютере.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

В сети существует ряд программ, предоставляющих похожий функционал, среди самых известных – Архивариус 3000, Copernic, FileSarchy. Однако, они не предоставляют не весь функционал. Ряд функционала, отсутствующий в программах:

1. Визуализация процесса индексирования
2. Прямое создание индекса
3. Поиск по смежным словам
4. Поиск по расширениям

Ввиду большого функционала, предоставляемого некоторыми программами (точная настройка морфологий, лемматизация и др.) скорость индексирования таких программ не является наилучшим показателем. Сравнения приведены ниже.

Данные: директория размером 170 мб., индексируемые форматы (\*.txt, \*.html, \*.xml), подстрока поиска «test», без поиска подстроки. Назначение директории – программные проекты (файлы \*.java, \*.db, \*.txt, \*.xml, \*.html, \*.class,…).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Архивариус 3000 | FileSearchy | Copernic | FileFinder (текущий проект) |
| Размер индекса | 3.33 мб | - | - | 9.68мб |
| Время индексирования | 26 сек. | >10 мин. (вся система) | 7 мин 44 сек (нет точного выбора форматов) | 12 сек |
| Время поиска | 0.01 сек | 2.44 сек. | 0.44 сек | 0.8 сек |

*Табл. 1.* Сравнение времени индексирования известных программ.

Сравнение использования системных ресурсов при индексировании и поиске.

FileFinder – текущий проект.

*Рис. 5.* Затраты ресурсов FileFinder при индексировании тестовой директории

Границами выделен фрагмент по времени, на котором шло индексирование директорий. На графике видно, что число записей на диск, как и число использованной оперативной памяти не выявило скачков, однако в силу того, что в задачи программы не ставился контроль загруженности центрального процессора, программа использует все доступные ресурсы, чтобы быстрее завершить индексирование. Скачок загрузки ЦП справа от границ – поиск. Он не вызывает серьёзных нагрузок.

Архивариус 3000.

Рис. 6. Затраты ресурсов Архивариус 3000 при индексировании тестовой директории.

Границами выделен фрагмент времени, на котором шла индексирование тестовой директории. На графике видно, что программа Архивариус 3000 не вызывает серьёзных затрат ЦП, однако, в конце индексирование записывает большой поток данных на диск. Значит, в программе настроен буфер перед записью на большое кол-во вхождений. При поиске программа также обращается к диску, что требует более серьёзных затрат ресурсов, чем программа FileFinder (текущий проект).

Другие представленные программы не не имеют интерфейса для явного создания индексов указанных директорий.

Проанализированные данные, позволяют создать следующую таблицу сравнения программ, расположенную ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | FileFinder | Архивариус 3000 | FileSearchy | Copernic |
| Явное создание индексов | да | да | - | - |
| Индекс по форматам файлов | да | да | - | да |
| Поиск по подстроке | да | - | да | да |
| Отложенное индексирование | - | да | - | да |
| Морфологический анализ | - | да | - | - |
| Индикация процесса индексирования | да | да | - | - |
| Индикация процесса поиска | да | да | - | да |
| Запоминание результатов поиска | да | да | - | - |
| Работает на UNIX системах | да | - | - | да |
| Распространяется бесплатно | да | - | да | - |

*Табл. 2.* Сравнение программ для поиска файлов

C:\Users\Александр\Downloads\Products.png

Рис. 7. Приблизительное сравнение наиболее распространённых программ с текущим проектом по времени индексирования и поиска.

Тем самым, программа позволяет выполнять поставленные перед ней задачи за небольшое время индексирования и среднее время поиска по отношению к другим известным программам. Кроме того, разработка программы на Java позволяет ей быть запущенной на UNIX системах, что большинство существующих прототипов не реализует. Отличием от многих похожих разработок также является свободное распространение программы (на бесплатной основе).

1. **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**
2. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. 5ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. Какие системные требования Java?. [Электронный ресурс] //URL: <https://www.java.com/ru/download/help/sysreq.xml> (дата обращения: 12.04.2017, режим доступа: свободный)
11. Datastructures and Algorythms for Indexing. [Электронный ресурс] //URL: <https://www.cl.cam.ac.uk/teaching/1314/InfoRtrv/lecture2.pdf> (дата обращения: 21.12.2016, режим доступа: свободный)
12. ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13. Каталог, общее описание. [Электронный ресурс] //URL: <https://en.wikipedia.org/wiki/Directory_(computing))> (дата обращения: 16.04.17, режим доступа: свободный)
14. Токен, словарь терминов. [Электронный ресурс] // URL: <https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%B5%D0%BD> (дата обращения 18.04.17, режим доступа: свободный)
15. Executor. [Электронный ресурс] // URL: <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html> (дата обращения: 18.04.17, режим доступа: свободный)
16. Design Patterns. // Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides – USA: Addison Wesley, 1994 – 416 с.

# **Приложение 1. Терминология**

Ниже приведён список терминов, применяемых в области и работе.

**Директория** - каталог, директория, справочник, папка — объект в файловой системе, упрощающий организацию файлов. Позволяют упорядочивать информацию в файловой системе.

**Индекс** – указатель в общем смысле слова. В данной работе – указатель от слова к файлам его содержащим. Основной способ построения быстро работающих поисковых программ, т.к. в идеальном случае сложность поиска по индексу , где k – количество разнообразных встречающихся слов, а поиска без индекса – , где n – количество файлов в директории поиска, а – длина i-ого файла. Как видно из формулы, на достаточно большом количестве файлов линейных поиск будет работать во много раз дольше, чем индекс. Существует много вариантов и структур данных для его построения, наиболее известный – обратный индекс.

**Расширение** – последовательность символов, добавляемых к имени файла (в MS Windows после символа «.»), служащих для идентификации типа файла программами.

**Делопроизводство** – ведение канцелярских дел.

**Запрос на поиск** – сущность в компьютерном пространстве (объект), содержащая всю необходимую для проведения поиска по индексу информацию (слово для поиска, индекс в котором будет производиться поиск, доп. параметра проведения поиска по индексу).

**Запрос на индексирование** - сущность в компьютерном пространстве (объект), содержащая всю необходимую для проведения индексирования информацию (индекс для записи промежуточных и конечных результатов, каталоги и файлы, доп. Параметры).

**Токен** – последовательность символов, в данной программе различаются токены вида слово и цифры, а также служебные символы.

**Индексирование** (индексирование) – процесс просмотра данных с целью добавления сведений в отдельную структура для последующего быстрого поиска.

**Хранилище индекса** – структура данных, позволяющая сохранять сведения о включениях слов в файлы во время работы программы и по её завершению.

# **Приложение 2. Основные алгоритмы**

1. Алгоритм линейного поиска по файлам

Сложность , где n – кол-во файлов в директории поиска, li – размер i-ого файла директории.

Описание: самый простой поиск, просматривает все файлы с целью поиска слова. В одиночной итерации быстрее любого поиска, основанного на индексации (т.к. необходимо сначала построить его, чтобы потом ещё в поиске использовать). Нет затрат на доп. память.

1. Алгоритм индексирования (без улучшений и обработки текста доп. алгоритмами)

Сложность: , где n – кол-во файлов в директории поиска, li – размер i-ого файла директории. Также добавляется сложность записи в хранилище индекса в зависимости от его типа. В b-tree (наиболее распространённый алгоритм баз данных) сложность вставки , где k – кол-во уникальных слов.

Описание: линейный просмотр файлов с целью записи сведений о включениях слов в них. Сведения могут изменяться в зависимости от вариаций алгоритма индексирования и улучшения связанных с лексическим разбором, взвешиванием и тд. Минимальные данные о включении – слово и файл, в котором оно встретилось.

# **Приложение 3. Основные обработчики**

1. Обработчик индексирования содержимого

Индексирование содержимого файлов – долгий процесс, поэтому останавливать поток просмотра файлов не выгодно. К тому же, в этом потоке может быть дополнительное распознание содержания папок. Процесс индексирования содержимого был вынесен в отдельные вызываемые функции (содержащие 1. запрос на индексацию, 2. путь файла) передаваемые потоку обработки содержимого. Он (встроенные механизм java.util.concurrent) создаёт n-ое количество потоков, выполняющие эти функции. Процесс индексации распараллеливается. Количество потоков в обработчике было выбрано 3 на основе сделанных измерений скорости записи данных в хранилище, т.к. в таком количестве они не мешают друг-другу занимая время процессора на переключение, а позволяют эффективно выполнять поставленную задачу.

1. Обработчик заданий на поиск и индексацию

Запросы на индексацию могут выполняться продолжительное время, поэтому чтобы не лишать пользователя возможности работы с программой на этот промежуток, запросы передаются обработчику. В отдельно взятый период времени может выполняться до 5 потоков обработки запросов, т.к. если число потоков больше – сильно снижается производительность их выполнения (увеличивается время их выполнения).

# **Приложение 4. Обоснование принятых решений**

1. Буфер записи в хранилище индекса

Для уменьшения количества транзакций к хранилищу, запись происходит не по одному включению, а сразу по множеству, но в пределах одной транзакции (в случае с БД). При получении вхождения, поток записывает его в структуру данных LinkedList, пока размер листа не будет превышать 3-ёх кратного размера BATCH параметра (через сколько операций происходит настоящая запись) базы данных. Batch параметр базы данных установлен на 130 исходя из проведённых измерений среднего времени выполнения 3 заданий на индексирования тестовой директории. Результат можно видеть на рисунке ниже (рис. 8). Стоит учесть, что чем меньше размер batch, тем меньше конфликтов будет возникать в базе данных при параллельной записи.

*Рис. 8.* Определение размера партии перед записью в базу данных опытным путём.

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в документе) | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| Изм. | Измененных | Замененных | Новых | Анулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |