УДК 515.142.33

РАЗРАБОТКА ИНДЕКСАТОРА СИСТЕМЫ УПРАВЛЕНИЯ ДОКУМЕНТАМИ

М.В. СТЕРЖАНОВ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220113, Беларусь

Поступила в редакцию 25 мая 2012 года

Описана реализация алгоритма индексирования, основанного на построении инвертированного файла, данных системы управления техническими документами. Для различных видов сортировки результатов поиска формируются отдельные индексные базы. Также описаны формат и назначение индексных файлов, составляющих индексную базу. Описывается архитектура программного модуля построения индексной базы.

Ключевые слова: индексатор, индексная база, система управления документами.

Введение

В последние годы резко повысилась актуальность обработки больших объемов разнородной текстовой информации. Существенно возрос интерес к классификации больших массивов документов и быстрому поиску в них нужной информации (см., например, [1–3]). Индексы представляют собой метод ускорения доступа к необходимым данным, позволяя, например, эффективно выполнять запросы вида «выбрать запись с данным идентификатором» (такие запросы называются точечными, так как им отвечает лишь одна точка в пространстве атрибутов) или «найти все товары, произведенные до 2012 года» (запрос по интервалу, которому соответствует интервал в пространстве атрибутов). Индексы являются избыточными структурами в том смысле, что они всегда могут быть восстановлены по основному информационному содержимому.

Индексатор − это набор алгоритмов, которые управляют индексной базой. В рамках данной статьи описывается индексатор системы управления техническими документами, работающий на основе модели инвертированных файлов [4] . Физически данные системы размещаются в таблицах БД. Индексные файлы размещаются на диске на сервере приложений. Описываемый в данной статье модуль индексации реализован при помощи языка программирования *Java*. Программно информация о документе представлена классом *Document*, содержащем в себе коллекцию полей (экземпляров класса *Field*). Класс *Field* соответствует полю таблицы БД. На вход модуль индексирования принимает список атрибутов, по которым впоследствии будет осуществляться поиск. Для целочисленных атрибутов можно указать нижнюю и верхнюю границы. Если значение атрибута не попадает в данный диапазон, то это значение не будет индексироваться и поиск по данному значению атрибута вернет пустое множество. Также имеется возможность указать различные виды сортировки результатов при поиске. Например, можно задать два вида сортировки найденных документов. Первый – сначала по названию документа, затем по автору. Второй – сортировать только по типу документов. Каждый вид сортировки осуществляется отдельным компаратором (от *comparator* − сравнивающее устройство). Компаратор получает из конфигурационного файла последовательность полей, по которым будет производиться сортировка. Для каждого вида сортировки будут создаваться индексные базы в отдельном подкаталоге.

Структура индексатора

Данные индексируются блоками. Размер блока индексации задается в конфигурационном файле. Каждый документ обрабатывается четырьмя обработчиками (*FieldWriter*, *DocumentWriter*, *SortWriter*, *TermWriter*). Результат индексирования отдельного документа сохраняется в буфере сегмента индексации. После полного заполнения буфера содержимое сегмента сохраняется в соответствующие индексные файлы на диске. После обработки всех документов созданные файлы сегментов объединяются.

Обработчик *FieldWriter* извлекает из документа подлежащие индексированию поля и формирует файл данных, являющийся каталогом полей. У каждого объекта класса *Field* обработчик получает название поля и сохраняет все уникальные имена полей в массив. После обработки блока данных информация о полях сохраняется в файл *FieldData*. Сначала записывается количество полей, затем записываются их названия. Пример размещения записей в файле *FieldData* приведен на рис. 1.

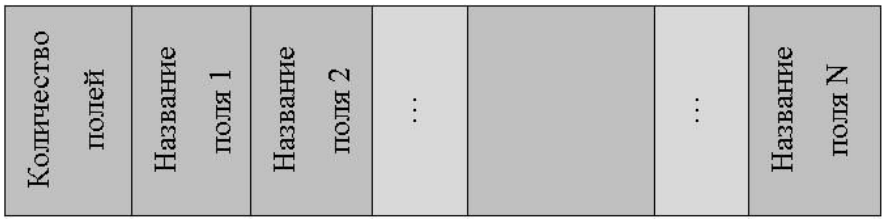

Рис.1. Структура файла *FieldData*

Обработчик *DocumentWriter* сохраняет информацию о всех документах в файл данных *DocumentData*, играющий роль каталога проиндексированных документов. Файл *DocumentData* позволяет осуществлять поиск, даже если исходные документы не доступны. В начале каждой записи указывается количество индексируемых полей. Затем для каждого поля указывается его идентификатор (соответствующий позиции в файле *FieldData*), является ли поле разбиваемым на лексемы, значение поля. Покажем структуру файла *DocumentData* на рис. 2.

Рис. 2. Структура файла *DocumentData*

Обработчик *SortWriter* отвечает за сортировку данных в файле *DocumentData*. При этом порядок размещения записей в файле *DocumentData* не меняется. Создается отдельный файл *DocumentIndex*, записи которого содержат указатели на документы *DocumentData*, расположенные в отсортированном порядке. Модуль индексирования может содержать несколько обработчиков *SortWriter*, которые отличаются только правилом сортировки данных. Интерфейс *FieldSortOrder* хранит массив названий полей, по которым будет осуществляться сортировка. Для каждого поля указывается направление сортировки (по возрастанию, по убыванию). Класс *FieldSortComparator* реализует интерфейс *FieldSortOrder*. Перед выполнением операции сравнения значения полей обрабатываемых документов помещаются в его внутренний массив *String values* [] в соответствии с порядком, определенным *FieldSortOrder*. Затем в методе *compareTo* последовательно попарно сравниваются элементы *values*[*i*] двух документов. Метод

compareTo будет использоваться внешним алгоритмом сортировки для сравнения двух документов.

Результатом работы обработчика *SortWriter* является создание файла *DocumentIndex*, содержащего позиции размещения в файле *DocumentData* отсортированных документов. Для каждого компаратора создается отдельный файл *DocumentIndex*.

Диаграмма классов, осуществляющих сравнение и сортировку документов, представлена на рис. 3.

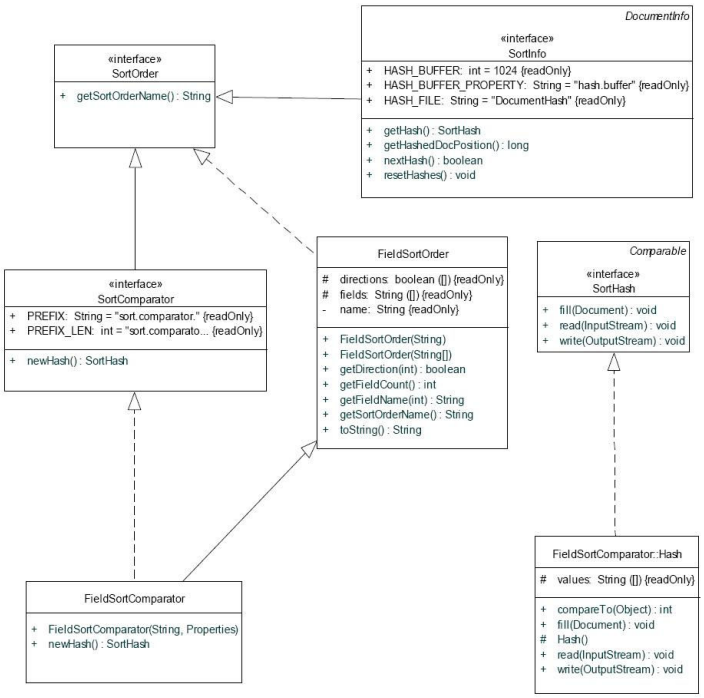

Рис. 3. Диаграмма классов *FieldSortComparator*

Обработчик *TermWriter* отвечает за построение лексикона − множества терминов, используемых в документах. Также учитывается частота вхождения терминов в документе.

Опишем вспомогательные классы *Term* и *TermDoc*. Класс *Term* хранит значение термина и имя поля, в котором данный терм был найден. Класс *TermDoc* содержит в себе два поля:

doc – номер обрабатываемого документа;

freq – число повторений термина в документе doc.

Очевидно, что класс *TermDoc* не содержит связи между документом и входящими в него терминами. Данная связь отражается в коллекции *TreeMap<Term*, *ArrayList<TermDoc>> terms*, в которой каждому термину ставится в соответствие множество номеров документов с указанием количества повторений данного термина в документе. Списки *ArrayList<TermDoc>* принято называть инвертированными списками или пост-листами [5].

Рассмотрим действия *TermWriter* при обработке документа:

1. Для всех индексируемых полей документа выполняется шаг 2.

2. Если поле помечено как «разбиваемое на лексемы», то обработчик извлекает из поля все лексемы и создает для каждой их них экземпляр класса *Term*, конструктору которого передаются в качестве параметров название поля и лексема. Если поле не помечено как «разбиваемое на лексемы», то сразу создается экземпляр класса *Term*, конструктору которого передаются в качестве параметров название и значение этого поля. После создания термин добавляется в коллекцию terms при помощи метода *addTerm*.

3. В методе *addTerm* сначала проверяется была ли найдена текущая лексема в уже обработанных документах. Если лексема встречается впервые, то в коллекцию *terms* добавляется текущий термин и новый *ArrayList<TermDoc>*, содержащий информацию о единичном вхождении термина в текущем документе. Если лексема встречалась ранее, то для текущего документа увеличивается счетчик количества повторений термина.

Блок-схема метода *addTerm* представлена на рис. 4.

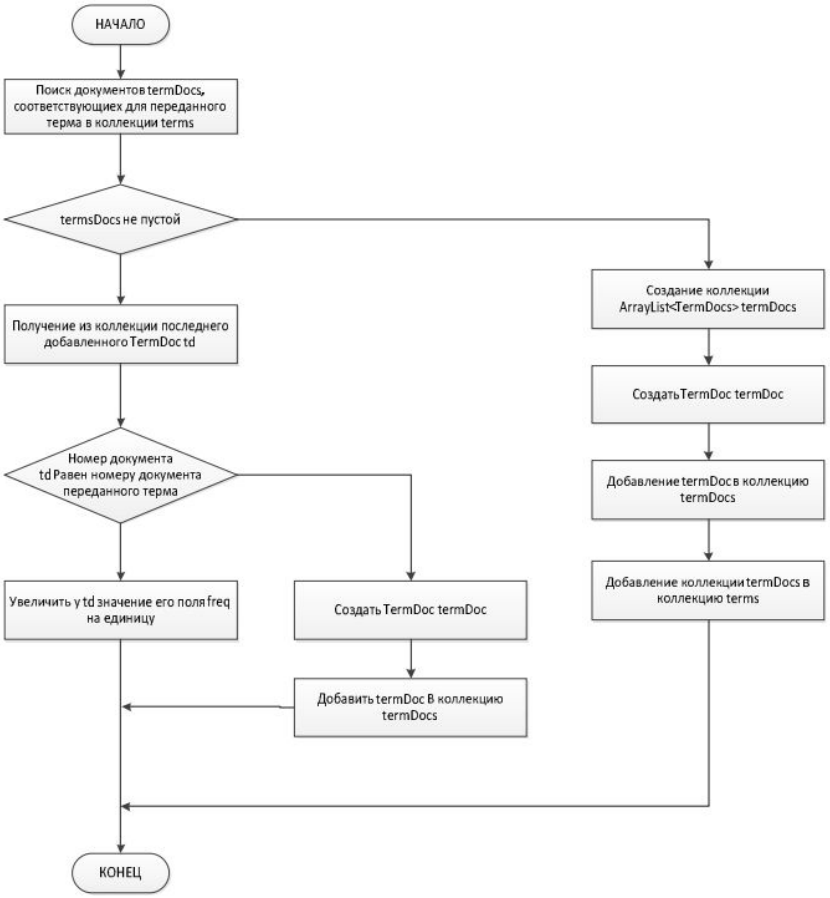

Рис. 4. Блок-схема метода *addTerm*

Рассмотрим процесс сохранения информации о терминах в индексную базу. Первой записью в файле *TermIndex* является количество сохраненных терминов. Затем последовательно следуют записи, хранящие описание терминов. Каждая запись содержит идентификатор поля (полученный при помощи *FieldWriter*), значение термина t, количество документов, содержащих в себе термин *t* (получается вычислением размера коллекции *ArrayList<TermDoc> TDocs*, соответствующей термину *t*), текущее смещение в файле *TermDocuments*. Файл *TermDocument* является вспомогательным файлом, хранящим указатели на записи из файла *DocumentIndex*, соответствующие документам, содержащим в себе термин *t*. При добавлении записи в файл *TermIndex* в качестве *TDOffset* записывается значение текущего смещения в файле *TermDocuments*. После сохранения термина *t* в файл *TermIndex* в файл *TermDocuments* сохраняются позиции размещения записей файла DocumentIndex, соответствующих документам, представленным в коллекции *TDocs*. В файл *DocumentFreqs* сохраняются значения частоты появления термина t в этих документах.

Покажем пример размещения информации о документах в индексной базе на рис. 5.

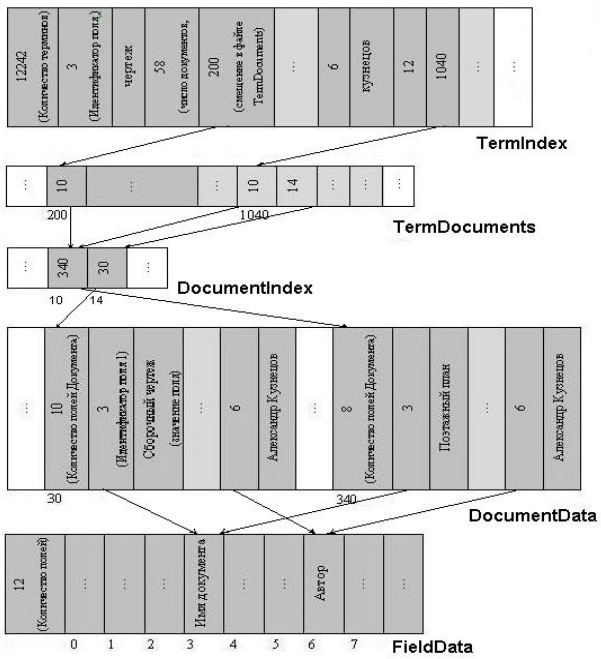

Рис. 5. Структура индексной базы

Заключение

Практическая значимость работы заключается в том, что описанный индексатор может найти применение не только в СУД, но и в различных справочно-информационных системах.

Особенностью предложенного решения является создание отдельных индексных баз с применением многоуровнего индекса для различных критериев сортировки результатов поиска, что ведет к сокращению времени поиска. Направлением дальнейших исследований является создание эффективных алгоритмов поиска документов на основе информации, накопленной в индексной базе.

Данная работа выполнялась при поддержке гранта Ф11-М210 Белорусского республиканского фонда фундаментальных исследований.

M.V. STERJANOV

INDEXER FOR DOCUMENT MANAGEMENT SYSTEM

Abstract

Indexation algorithm based on inverted file pronciple is presented. Algorithm creates separate index databases for each sorting type. Format and meaning of every constituent of index database is detailed. Also, program architecture of developed module is described.

Список литературы

1. *Witten I.H., Moffat A., Bell T.C.* Compressing and Indexing Documents and Images. San Diego, CA, 1999.
2. *Веретенников А.Б.* Программный комплекс и эффективные методы организации и индексации больших массивов текстов: дисс. канд. физ.-мат наук. Екатеринбург, 2009
3. *Weber I.* Efficient index structures for and applications of the CompleteSearch engine. Saarland University. 2007.
4. *Zobel J., Moffat A., Ramamohanarao K.* // ACM Trans. on Database Systems. 1998, Vol. 23, Issue 4. P. 453−490.
5. *Faloutsos C., Douglas W.* A survey of information retrieval and filtering methods. 1995.