УДК 681.3

ПОСТРОЕНИЕ ПОИСКОВЫХ INTERNET/INTRANET СИСТЕМ НА ОСНОВЕ ПОЛНОТЕКСТОВЫХ БАЗ ДАННЫХ

Ю.И. ВОРОТНИЦКИЙ, А.В. ПУПКО

Предлагаются основные алгоритмы создания поисковых баз данных, содержащих полнотекстовые копии документов, размещаемых в Internet/Intranet сетях. Алгоритм индексирования документов строится по принципу формирования числового указателя. Рассмотрены алгоритмы отсечения, используемые при создании поисковых баз данных.

В условиях экспоненциального роста объемов информации, размещаемой в Internet и корпоративных Intranet сетях, проблема быстрого и релевантного поиска в них приобретает все большую актуальность [1].

Рассмотрим основные алгоритмы создания поисковой базы данных (ПБД), содержащей полнотекстовые копии документов, размещаемых в Internet/Intranet сетях. Предлагаемые алгоритмы построены на следующих предположениях:

1) любой документ в ПБД можно описать посредством конечного набора числовых идентификаторов (индексов), что предполагает возможность описания документа с помощью набора слов, которые могут быть преобразованы в числовую форму;

2) для осуществления поиска в формируемой ПБД используются возможности стандартных SQL-запросов по числовому представлению документов;

3) полные тексты индексируемых документов хранятся в ПБД, причем для их отображения также используются возможности стандартных SQL-запросов к реляционным БД.

Подход к построению ПБД, основанный на хранении в них копий полнотекстовых документов, позволяет существенно сократить время доступа к текстам, что особенно актуально в условиях ограниченной пропускной способности каналов доступа в Internet. Кроме того, предлагаемый подход обеспечивает доступ к текстам документов в случае отказа сервера, где расположена исходная информация, повреждения или непреднамеренного кратковременного удаления исходного документа. В этом случае текстовая информация может быть получена из копии, хранящейся в ПБД, обновление информации в которой будет происходить при плановых переиндексациях документов. При этом объемы хранимой текстовой информации для корпоративных и даже общенациональных поисковых систем не представляются слишком большими. Так, общий объем текстовых документов, хранимых в белорусской зоне Internet, оценивавшийся нами при построении реальных поисковых систем [2], не превышает 15 Гбайт.

По сути, вместо традиционной двухуровневой модели организации поиска, при которой пользователь на первом уровне осуществляет поиск по

индексам и просмотр ключевых слов, а на втором — просматривает исходный документ на «родительском» сервере (на это могут потребоваться значительные затраты времени, особенно в условиях ограниченной скорости доступа), предлагается новая трехуровневая модель организации поиска. В этой модели на первом уровне осуществляется поиск по индексам, на втором — просмотр копии документа в локальной ПБД, и только на третьем — просмотр исходного документа на «родительском» сервере. Это позволяет пользователю на втором уровне организовать быстрый первичный анализ документов, с целью отсечения документов и сайтов, не содержащих требуемой информации, что отчасти облегчает проблему поиска в «зашумленном» Internet.

Сам процесс индексирования документа можно представить следующей последовательностью действий:

1) получение документа;
2) вычленение слов;
3) отсечение неинформационных слов;
4) вычисление числовых идентификаторов слов (индексов);
5) занесение в ПБД информации о документе; его индексов и полного текста.

При такой схеме построения и обновления ПБД основные затраты времени приходятся не на поиск, а на обработку документа, которая осуществляется не в реальном времени выполнения запроса, а один или несколько (в случае изменения исходного документа) раз при индексировании в системе. Архитектура ПБД, позволяющая реализовать данную схему, представлена на рис. 1., где

DOCUMENT — таблица, хранящая базовые характеристики документа, такие как адрес (url), размер (size), дата создания, дата индексирования, текущий статус и т.д.;

INDEX — таблица, хранящая индексы слов (fk_word), вес индексов (weight) и числовые идентификаторы документов (fk_document);

DICTIONARY — таблица, содержащая базовые слова (word) и их числовые идентификаторы (id_word);

DICTIONARY_NOT — таблица, содержащая слова (word) и их числовые идентификаторы (id_word), отсутствующие в таблице DICTIONARY, причем множества числовых идентификаторов этих двух таблиц не пересекаются;

DICTIONARY_NOISE — таблица, содержащая «шумовые» слова (word), которые требуют обязательного отсечения при индексировании.

Связь A между таблицами DOCUMENT и INDEX означает «ноль или больше», т.е. любой документ может не содержать или содержать индексы. Связи B1 и B2 между таблицами DICTIONARY, DICTIONARY_NOT и INDEX также означают «ноль или больше», т.е. любое слово может входить или не входить в описание документа.

Количество записей C в таблице INDEX можно вычислить по формуле

$$C = \sum_{1}^{n} S_n \ , \qquad (1)$$

где n — количество документов; S_n — количество индексов для документа n.

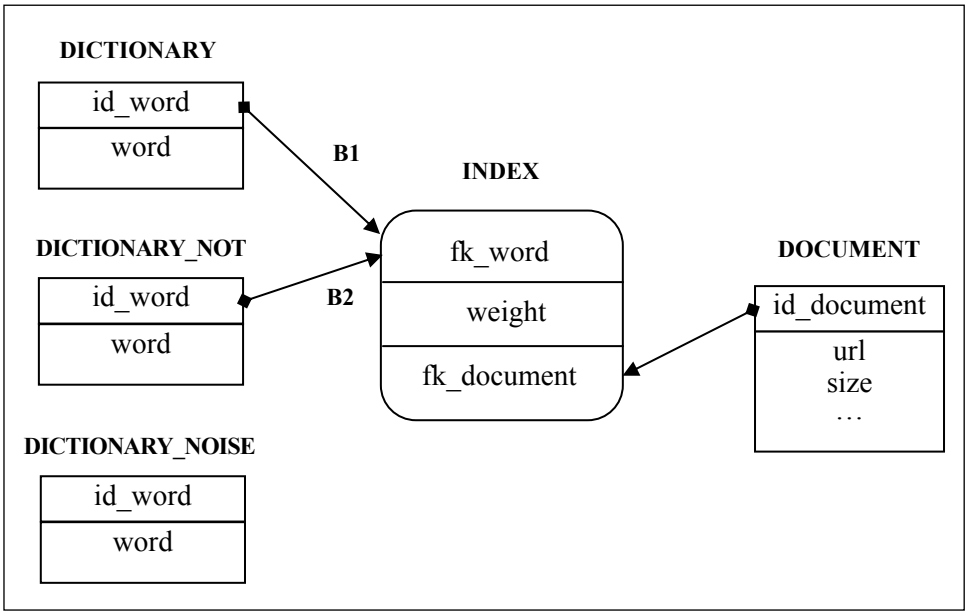

Рис. 1. Схема базы данных поисковой системы.

Как уже было сказано, таблица индексов состоит из трех полей: fk_word fk_document типа int (4 байта) и weight типа smallint (2 байта). Общий размер таблицы $S = C \cdot 10$. Например, для 500 000 документов со средним количеством индексов, равным 250, объем таблицы составит около 1,25 Гбайт.

Правила построения таблицы индексирования можно разделить: общие правила (для всех документов) и специфические (для документов с особой структурой).

При вычленении слов из документа мы исходим из следующих общих правил:

1) слово может состоять только из таких наборов символов: прописных и строчных букв английского, русского/национального алфавитов; арабских цифр и символа «—» — тире или дефис;

2) символ, не принадлежащий к перечисленным выше, является разделителем;

3) слово может начинаться с буквы или цифры;

4) слово не может быть длиннее 42-х и короче 3-х символов;

5) слово не может начинаться или заканчиваться символом тире, а также содержать больше одного символа «—»;

6) в слове не могут одновременно присутствовать символы английского и русского/национального языка;

7) если в слове одновременно присутствуют символы английского и русского/национального языка, а после замены английских букв *C* и *I* на буквы русского и национального (например, белорусского) языка, мы получаем слово только с символами русского/национального языка, то последнее слово является корректным.

Очевидно, что для реализации соответствующего этим правилам алгоритма вычленения слов, БД, и ее таблицы, отвечающие за анализ слов, не требуются. Таким образом, данную часть алгоритма построения ПБД можно реализовать на языке высокого уровня, например *C*++.

Полученная информация о словах обрабатывается с помощью алгоритмов морфологического анализа и алгоритмов отсечения на уровне ПБД. В укрупненном виде эти алгоритмы описываются следующей последовательностью действий:

1) из рассматриваемого набора исключаются неинформативные слова, перечисленные в таблице DICTIONARY_NOISE (например, слова «для», «при», «and», «www», «index» и т.п.);

2) на основании таблицы DICTIONARY проводится морфологический анализ. Таблица DICTIONARY построена таким образом, что слова, отличающиеся друг от друга окончаниями, имеют одинаковый числовой идентификатор. Например, слова «самолет», «самолету», «самолетом» имеют одинаковый числовой идентификатор. Кроме того слова, которые при склонении изменяют корневую составляющую (например, неправильные глаголы английского языка), также имеют одинаковый идентификатор;

3) если слово не определено в пп. 1 и 2, то его нужно искать в таблице DICTIONARY_NOT. При его отсутствии и в этой таблице, слово заносится в нее с новым числовым идентификатором.

Заметим, что занесение в ПБД индексов для слов, которые были получены из таблицы DICTIONARY_NOT, является обязательным, поскольку это могут быть имена собственные, географические названия, аббревиатуры, коды изделий и т.д., то есть те слова, по которым чаще всего и осуществляется поиск.

При анализе слов, содержащих символ «—» (тире или дефис), последовательно рассматриваются следующие варианты:

1) анализируется слово, полученное в результате удаления символа «-», то есть, этот символ рассматривается как знак переноса; если данное слово найдено в таблице DICTIONARY или DICTIONARY_NOT, дальнейший анализ прекращается;

2) анализируется исходное слово (предполагается, что оно действительно содержит дефис); если данное слово найдено в таблице DICTIONARY_NOT дальнейший анализ прекращается;

3) анализируются слова, стоящие слева и справа от символа тире; если данные слова не найдены в таблице DICTIONARY или DICTIONARY_NOT, они заносятся в таблицу DICTIONARY_NOT с присвоением им соответствующих числовых идентификаторов.

Непосредственное занесение полученных значений индексов в таблицу INDEX, в рассматриваемой схеме формирования ПБД неизбежно приведет к тому, что обработка одного документа может достичь неприемлемой величины (при 30 000 000 записей в БД INDEX — до 30–50 мин для однопроцессорного сервера на базе процессора Intel Pentium 4 с объемом оперативной памяти 256 Мбайт и дисковом RAID массиве 5-го уровня объемом памяти 40 Гбайт). Это связано с тем, что для обеспечения поиска требуется наличие кластерного индекса по полю fk_word, и для любого

изменения в таблице INDEX необходима ее полная физическая пересортировка. Для оптимизации процедуры записи индексов можно использовать следующий алгоритм, требующий создания двух дополнительных таблиц ChangeDocument и ChangeIndex:

1) идентификаторы всех индексируемых документов заносятся в таблицу ChangeDocument;

2) все индексы документов заносятся в таблицу ChangeIndex;

3) периодически, в определенные моменты времени (например, один раз в сутки — ночью), выполняется следующая последовательность операций:

3.1. ПБД переводится в монопольный режим;

3.2. удаляется кластерный индекс по полю fk_word таблицы INDEX;

3.3. создается некластерный индекс по полю fk_document таблицы INDEX;

3.4. из таблицы INDEX удаляются все записи, соответствующие записям из таблицы ChangeDocument;

3.5. все записи из таблицы ChangeIndex переносятся в таблицу INDEX;

3.6. строится кластерный индекс по полю fk_word таблицы INDEX;

3.7. очищаются таблицы ChangeDocument и ChangeIndex;

3.8. ПБД переводится в рабочий режим.

Использование морфологической БД позволяет при поиске информации использовать упрощенный алгоритм поиска орфографических ошибок, который базируется на том, что ошибка или опечатка может возникнуть только в трех случаях:

4) неправильно набран символ;

5) символ пропущен;

6) вставлен лишний символ.

Просматривая слово в различных вариантах по таблице DICTIONARY, мы можем определить наиболее близкий эквивалент. К сожалению, данный метод нельзя использовать при автоматическом индексировании документов. Это объясняется неоднозначностью определения эквивалента слова, а также процессом выполнения анализа имен собственных. Например, аббревиатура БГУ (Белорусский государственный университет) будет рассматриваться как БОГУ или БЕГУ.

Отметим, что существует альтернативная схема морфологического анализа, предполагающая, что таблица DICTIONARY разбивается на две таблицы: корни слов и окончания. Однако несомненный выигрыш в объеме хранимой информации в этом случае приводит к появлению таких недостатков:

1) усложнению алгоритма поиска слова;

2) усложнению поиска опечаток;

3) так же как и в рассмотренном ранее способе анализа обязательно перечисление слов, которые при склонении изменяют корневую составляющую;

4) быстродействие при поиске не выше, а иногда и ниже чем в описанном ранее способе морфологического анализа.

Отсутствие выигрыша в скорости поиска обусловлено тем, что в БД (если таблицы отсортированы) поиск проводится по *B*-дереву. Для поиска в 130 000 записях (различные корни слов русского/английского языка) требуется около 17 проходов, а при поиске в 980 000 записях (различные словоформы русского/английского языка) требуется 20 проходов. Но при этом таблицу в 130 000 записей придется просматривать несколько раз, находя подходящее окончание.

Рассмотренная нами последовательность алгоритмов обеспечивает быстрый, но не релевантный поиск. Релевантность может быть обеспечена тем, что при формировании индексов конкретного документа будут сохраняться не только индексы различных слов, но и частоты их вхождения в документ [3]. Тогда при выполнении поиска можно будет упорядочить его результаты по частоте вхождения искомого слова в документ. Кроме того, можно сохранять не абсолютную, а относительную частоту вхождения и ввести некоторое пороговое правило, которое будет использоваться в качестве критерия отбора документа [4].

Предлагается при формировании значения частоты вхождения слова в документ умножать число вхождений на весовые коэффициенты, зависящие от того, в каком разделе документа встретилось слово. Например, если слово присутствует в заголовке документа или в его описании (ключевые слова), весовой коэффициент выбирается порядка 10 000. Однако если такое слово не встретится ни разу в основном тексте документа, оно вообще исключается из анализа. Этим отсекается шум, неизбежно возникающий при некорректном формировании заголовка и описании документа (например, таким образом недобросовестные администраторы пытаются повысить рейтинг документа в поисковых системах).

ЛИТЕРАТУРА

1. *Мелюхин И.С.* Информационное общество: истоки, проблемы, тенденции развития. — М.: Изд-во Москов. ун-та, 1999.
2. *Пупко А. В., Воротницкий Ю. И.* Архитектура поисковой системы по белорусским информационным ресурсам в Internet // Проблемы проектирования информационно-телекоммуникационных систем. – Мн.: Изд-во Белорус. ун-та, 2001. — С. 72–78.
3. *Моисеенко С., Майстренко А.* Релевантность полнотекстового поиска: подход на основе построения терминологической базы документов // http://www.citforum.ru/database/articles/.
4. *Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В.* Базы данных. Интеллектуальная обработка информации. — М.: Нолидж., 2000.

Поступила 15.04.2002