DOI: 10.3745/KIPSTA.2010.17A.5.229

다중 키워드 검색에 적합한 동등조인 연산 결과의 동적 관리 기법

임 성 채†

요 약

인터넷이나 기업체 안에서 생성되는 문서의 수가 빠르게 증가하고 있고 이에 따라 효율적인 문서 검색 서비스의 중요성도 함께 커지고 있다. 이런 검색 환경에서 사용자의 검색 질의를 미리 예측할 수 없기 때문에 문서 내의 키워드를 자동 추출하여 색인어로 사용하는 전문검색(full-text search)이 일반적으로 적용된다. 전문검색을 위해 생성된 색인 파일의 크기는 문서 수 증가로 대용량화 되고, 이런 대용량 색인에 대한 다중 키워드 질의 처리에는 과도한 디스크 비용이 초래될 수 있다. 논문에서는 이런 비용 문제를 해결하기 위해 대용량 문서의 전문검색 시스템에서 다중 키워드 질의를 효율적으로 처리할 수 있게 하는 색인 파일 구조 및 관리 기법을 제안한다. 제안된 방법은 다중 키워드 검색에 적합한 것으로 알려진 역파일을 기본 색인 구조로 하며, 질의 처리의 조인 연산과 랭킹 연산에 적합하도록 색인 파일을 계층화한다. 이를 바탕으로 다중 키워드 질의를 구성할 확률이 높은 키워드 쌍에 대한 조인 연산 결과를 주기억장치 공간에 동적으로 저장함으로써 디스크 사용량을 크게 줄일 수 있다. 논문에서는 제안된 기법의 우수성을 보이기 위해 디스크 비용 모델에 기반한 성능 비교도 수행한다.

키워드 : 검색엔진, 웹 검색, 역파일, 색인파일

Dynamic Management of Equi-Join Results for Multi-Keyword Searches

Sung Chae Lim†

ABSTRACT

With an increasing number of documents in the Internet or enterprises, it becomes crucial to efficiently support users' queries on those documents. In that situation, the full-text search technique is accepted in general, because it can answer uncontrolled ad-hoc queries by automatically indexing all the keywords found in the documents. The size of index files made for full-text searches grows with the increasing number of indexed documents, and thus the disk cost may be too large to process multi-keyword queries against those enlarged index files. To solve the problem, we propose both of the index file structure and its management scheme suitable to the processing of multi-keyword queries against a large volume of index files. For this, we adopt the structure of inverted-files, which are widely used in the multi-keyword searches, as a basic index structure and modify it to a hierarchical structure for join operations and ranking operations performed during the query processing. In order to save disk costs based on that index structure, we dynamically store in the main memory the results of join operations between two keywords, if they are highly expected to be entered in users' queries. We also do performance comparisons using a cost model of the disk to show the performance advantage of the proposed scheme.

Keywords : Search Engine, Web Searches, Inverted Files, Index Files

1. 서 론

웹 문서 및 기업체 문서 검색에 주로 이용되는 전문검색(full-text search) 기능은 문서 내의 전체 키워드를 색인한 후, 검색자의 질의에 포함된 키워드와 비교하여 검색 관련성이 높은 문서들을 선택해 주는 기능이라 할 수 있다[1-3]. 사용자는 찾고자 하는 문서와 연관성이 높다고 생각하는 키워드를 두 개 이상 입력할 수 있으며, 이런 검색 질의를 다중 키워드 질의라 부른다. 검색 서비스 사용자는 자신이 원하는 문서를 발견할 때까지 여러 가지 다른 키워드 조합을 반복적으로 입력할 수 있음으로 효과적인 전문검색 시스템이라면 다양한 키워드로 구성된 다중 키워드 질의를 실시간으로 빠르게 처리 할 수 있어야 한다[3].

다중 키워드 질의를 처리할 때 소요되는 디스크 비용은 색인된 문서 수가 증가함에 따라 빠르게 커지는 경향이 있다. 예를 들어 웹 검색이라면 검색 대상이 되는 문서 수가

† 종신회원 : 동덕여자대학교 컴퓨터학과 조교수
논문접수 : 2010년 4월 26일
수 정 일 : 1차 2010년 6월 18일
심사완료 : 2010년 6월 24일

수 천만 개를 넘는 것이 일반적이며, 이 경우 전문검색을 위한 색인 데이터 크기는 수백 기가 바이트를 넘게 된다[4-6]. 이런 큰 규모의 색인 데이터를 이용하여 키워드 조인(join) 연산과 문서 관련성을 따져 보는 랭킹 연산을 수행해야 하기 때문에 상당한 디스크 대역폭 사용이 필요하다. 이런 대용량의 색인 파일을 효과적으로 다룰 기법이 없다면 적절한 질의 응답시간을 보장할 수 없고 디스크 비용이 커져 검색 시스템 구축에 큰 장애 요소가 될 수 있다[6].

본 논문에서는 이런 문제 인식에 따라 대용량 문서의 전문검색 시스템에서 다중 키워드 질의를 효과적으로 처리할 수 있도록 하는 색인 파일의 동적 관리 기법을 연구한다. 제안된 방법은 다중 키워드 검색에 적합한 것으로 알려 진 역파일 색인 구조를 기본 색인 구조로 하고 있으며, 질의 처리 과정에 요구되는 조인 연산과 랭킹 연산에 적합하도록 색인 파일을 계층화함으로써 이 두 단계의 연산에 적합한 색인 파일 구조를 제시한다. 이를 바탕으로 주기억장치 공간에 다중 키워드 질의에 자주 포함되는 키워드 쌍에 대한 조인 연산 결과를 저장함으로써 전체 디스크 사용 비용을 크게 줄일 수 있는 색인 파일의 동적 관리 기법을 제안한다. 그리고 제안된 기법에 따라 감소되는 디스크 비용을 계산하기 위해 질의 처리시의 디스크 비용 모델을 제시하고 이를 통해 성능 향상 결과를 보인다.

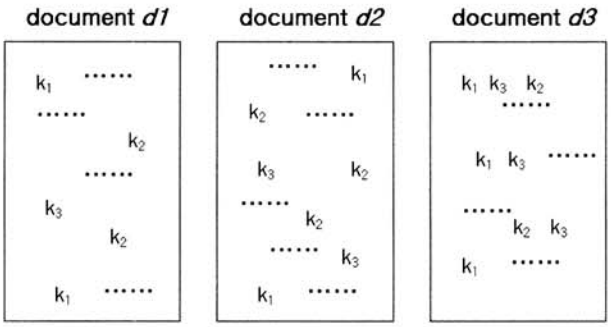
논문의 구성은 다음과 같다. 2장에서는 제안 기법의 설명을 위해 다중 키워드 검색에 대한 배경 지식을 소개하고 기존 연구들에 대해 알아본다. 3장에서는 제안한 색인 파일 구조 및 동적 색인 파일 관리 기법을 적용한 질의 처리 알고리즘을 소개한다. 4장에서는 기법 적용에 따른 성능 개선 효과를 보이고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 색인 파일

국내 도서관이나 논문 검색 시스템에서는 키워드 집합을 사람이 결정하고, 한정된 색인 키워드 집합에 대한 질의를 처리하는 것이 일반적이다[2]. 하지만 웹 검색과 같이 사용자의 질의를 예측하기 어렵고 색인 할 문서 수 역시 빠르게 증가하는 환경에서는, 전문검색(full-text search)이 사용된다. 즉, 색인 키워드를 사람이 통제하지 않고 문서 내에 출현한 모든 키워드 집합을 색인한다. 이 때 색인 기법은 크게 시그내쳐(signature)를 사용한 기법이나 역파일(inverted file)을 생각해 볼 수 있다. 이 중에서 질의 관련도(relevancy)를 결정하는데 유리한 역파일 구조가 전문검색에 사용되고 있다[5].

역파일은 문서의 모든 키워드(혹은 색인어)를 해당 문서의 식별자(identifier)와 함께 추출한 후 추출된 키워드 별로 출현 문서 식별자를 기록하는 방식이다. 예를 들어, 세 개의 문서 D_1, D_2, D_3에 $D_1[k_1, k_2, k_3]$, $D_2[k_2, k_3]$, $D_3[k_1, k_3]$와 같이 키워드가 출현하고, 이들 문서의 식별자를 d_1, d_2, d_3라 할 때, 역파일은 $k_1(d_1, d_3)$, $k_2(d_1, d_2, d_3)$, $k_3(d_1, d_3)$와 같이 키



(그림 1) 문서 내 키워드 출현 예

워드 별로 식별자를 기록한다. 논문에서는 이런 키워드 별 식별자 정보를 문서식별자 리스트라 부른다. 모든 색인된 키워드에 대한 문서식별자 리스트로 역파일을 구성한 후 동등조인(equi-join)을 수행함으로써 주어진 키워드를 모두 포함한 문서를 찾아 낼 수 있다. 이런 색인 파일에는 랭킹 연산에 사용될 추가 색인 정보도 함께 저장된다.

이런 추가 정보 중 필수적인 것으로 키워드의 문서 내 위치 정보가 있다. 예를 들어 (그림 1)에서와 같이 다중 키워드 질의 $Q = \{k_1, k_2, k_3\}$에 대한 동등 조인 결과로 다음 세 개 문서가 있다고 하자. 이 경우 이 세 문서의 질의 연관도는 $d1 < d2 < d3$의 순으로 추정해 볼 수 있다. 이는 문서 내에 나타난 키워드 간의 인접도나 빈도들을 참고한 결과이다. 이런 위치 정보와 함께 다른 색인 정보도 요구된다. 웹 문서의 키워드의 웹 문서 타이틀 존재 여부, 폰트 스타일이나 크기, Url 내의 존재 유무, 테이블 제목인지의 여부, 앵커 텍스트 위치의 여부 등 매우 다양한 정보가 있을 수 있다[5-7]. 이런 정보들이 함께 저장되기 때문에 색인 파일의 크기는 HTML 태그(tag)들을 삭제한 상태의 웹 문서의 크기보다 1.5 배 정도까지 커질 수 있으며 5,000만 건 정도의 웹 문서를 색인 했을 때 실제 150기가 바이트를 상회하는 정도의 색인 파일이 생성되었다. 이런 크기는 색인 문서 수가 늘어남에 따라 함께 커진다.

2.2 기존 연구

전문검색에 있어 질의 처리 효율성을 높이고자 하는 연구는 주로 웹 검색 분야에서 활발하였다. 이는 웹 검색 서비스의 색인 문서 수가 매우 크고 그 수 또한 빠르게 증가하고 있기 때문이다. 국내 존재하는 인터넷 사이트의 경우 블로그나 카페 사이트를 제외하고도 HTML 문서 수가 5000만 페이지를 상회하며, 구글 검색 사이트의 경우 색인 문서 수가 수십억 개를 넘고 있다[1, 4]. 이렇게 큰 규모의 전문검색 시스템에서는 다중 키워드 질의를 처리하는 비용을 줄이는 것이 매우 중요하다. 단일 키워드 검색의 경우는 읽어 들일 색인 파일이 크기가 다중 키워드 질의에 비해 매우 작고 랭킹 연산에 드는 비용도 크지 않다. 더욱이 랭킹 연산 결과를 색인 파일에 미리 저장한 후 정렬해 둠으로써 질의 처리 비용을 크게 줄일 수도 있다. 이에 비해 다중 키워드 질의의 경우 처리 비용을 줄일 수 있는 방법이 간단치 않다.

다중 키워드 질의 처리 비용을 줄이기 위한 연구로 질의 처리 결과를 디스크나 주기억장치에 저장하였다가 재사용하는 방식이 있다. 이런 연구에서는 결과 페이지를 저장할 질의 집합을 어떤 정책에 따라 정할 것인지, 어떤 서버에 저장할 것이지, 주기억장치와 디스크 장치 중 어디에 저장할 것인지 결정하는 것이 중요한 연구 대상이 된다[6, 7, 11, 13]. 또한 어떤 문자열 찾기 방식을 사용하여 랭킹 연산의 키워드간 인접도를 계산할 지에 대한 연구도 존재한다[17]. 이런 연구는 상용 웹 검색의 경우 초당 처리할 사용자 질의가 매우 많기에 CPU 비용이 빠르게 증가하는 것을 막기 위해 필요하다.

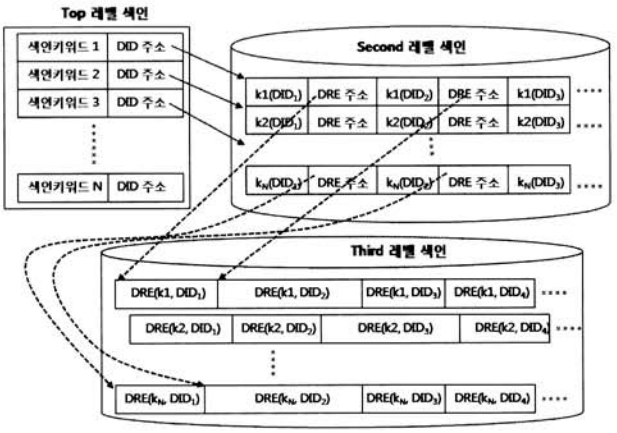
웹 서버에 입력되는 질의들은 로그 데이터에 기록되며 이 로그 데이터를 사용하여 자주 입력되는 질의를 관리하고, 여기서 얻어 진 정보를 사용하여 캐시할 질의 처리 결과를 결정하는 방식이 있다[10, 12, 13, 15]. 이런 방식은 포탈들에 의해 널리 사용되고 있다. 하지만 모든 질의에 대한 처리 결과를 저장할 수는 없기 때문에 사용자 질의를 실제로 처리해야 할 상황도 자주 발생한다. 따라서 질의 처리 비용을 감소시킬 수 있는 방안은 계속 요구된다고 하겠다. 본 논문의 연구는 이처럼 실제 질의를 처리해야 할 상황에서 가급적 적은 디스크 비용 만으로 동등조인 연산 및 랭킹 연산을 수행할 수 있는 기법에 대한 것이다.

앞에서와 같이 검색 처리 결과 자체를 저장하는 기법 외에도 다중 키워드 질의 처리에 읽혀질 색인 파일의 크기를 줄임으로써 질의 처리 비용을 줄이고자 하는 연구가 있다. 이런 연구로 색인 파일의 상당 부분을 차지하는 문서식별자 리스트를 압축 저장한 후에 동등조인을 수행하기 전에 압축 해제하여 사용하는 방식이 있다[16, 17]. 이런 방식을 통해 디스크로부터 색인 파일을 읽어 들이는 비용은 감소하지만, 주기억장치로 압축을 풀어야 하기에 CPU 비용이 증가하고 줄일 수 있는 색인 파일 크기에서도 한계가 있다. 또 다른 방식은 색인 파일을 하나의 디스크에 저장하는 경우 읽기 속도가 느릴 수 있기에 여러 개의 디스크를 사용하여 질의 응답 시간을 높이기 위한 연구도 있다[8, 9, 11]. 이와 같은 색인 파일 크기를 줄이거나 여러 디스크에 분산시키는 방식은 논문의 기법과 서로 상충되지 않기에, 함께 적용 가능하다.

3. 제안하는 색인 관리 기법

3.1 제안하는 색인 구조

제안하는 색인 파일 구조를 설명하기 위해 다중 키워드 질의 처리 과정을 먼저 살펴본다. 임의의 질의 $Q = \{k_1, k_2, k_3\}$가 세 개 키워드로 구성되고, 이들 키워드에 대한 문서식별자 리스트를 L_1, L_2, L_3라고 하자. 질의 Q는 2단계로 처리될 수 있다. 첫 번째 단계는 세 개 키워드에 대한 동등조인 수행 단계이다. 즉, L_1, L_2, L_3를 읽어 들여 공통의 문서 식별자를 선택함으로써 k_1, k_2, k_3를 모두 포함하는 문서 집합 D를 구한다. 다음 단계에서는 D에 속한 문서의 질의 관련성을 추정하는 랭킹 연산 단계이다. 이 단계에서는 앞서 언급한 문서 내 위치 정보 및 추가 색인 정보들이 사용된다.



(그림 2) 제안된 계층적 색인 구조

논문에서는 동등조인 단계에 사용될 문서식별자 리스트를 DID(Document ID), 랭킹 단계에서 사용될 데이터를 DRE (Data for Rank Evaluation)라고 부른다. DRE는 키워드의 문서 위치 정보와 색인 메타(META) 정보로 구성된다. 여기서 메타 정보는 앞서 언급한 문서 중요도를 암시하는 추가 색인 정보를 의미한다. DID와 DRE 데이터는 질의 처리 과정 중 서로 다른 단계에서 사용되며 디스크 상에 서로 인접 위치할 필요는 없다. 또한, DID 데이터는 키워드 별로 전체 데이터가 동등조인 단계에서 읽혀 져야 하지만 DRE의 경우는 동등조인 결과로 선택된 문서에 대해서만 선택적으로 읽을 수도 있다.

이런 DID와 DRE 데이터의 차이점 및 질의 처리 과정을 고려해서 (그림 2)와 같은 구조의 색인 파일을 적용한다. 그림의 최상위(top) 색인 계층은 검색 시스템의 시작과 함께 주기억장치에 위치하며 키워드 별로 해당 키워드의 DID 데이터가 위치한 디스크 정보가 주어진다. 즉, DID 데이터 위치를 저장하는 일종의 디렉터리 데이터이다. 바로 아래 두 번째 계층에 DID 데이터 파일이 존재하며 이 곳에 문서식별자 정보와 관련 DRE 데이터의 디스크 위치 정보가 저장된다.

그림의 DID 데이터에 $k_x(DID_y)$로 표시된 사각형은 키워드 k_x가 존재하는 y번째 문서의 문서 식별자 값을 나타내며, 바로 다음의 사각형은 식별자 $k_x(DID_y)$의 문서에 대한 k_x의 DRE 데이터 주소값이다. 즉, 두 번째 계층의 색인 파일에는 출현 문서의 식별자 값과 관련 DRE 데이터의 위치 정보가 한 쌍으로 저장되며, 이런 쌍은 키워드가 출현한 문서 수 만큼 생성된다. 한편 세 번째 계층은 DRE 색인 파일이며 사각형 표시된 DRE(k_x, y)는 키워드 k_x가 문서 식별자 $k_x(DID_y)$의 문서 내에서 가지는 위치 정보 및 메타 색인 정보이다. DRE 데이터도 키워드 별로 한번에 읽혀 질 수 있기에 동일한 키워드 별로 디스크에 서로 인접시킨다.

제안한 색인 파일을 사용해서 키워드 k_1과 k_2를 가지는 질의를 처리하는 과정을 설명해 본다. 먼저 주기억장치에

위치한 최상위 색인 데이터를 참조하여 두 번째 계층 파일에서 k_1과 k_2 각각의 데이터를 읽어 들인다. 그리고 이곳에 기록된 DID 데이터를 가지고 동등조인을 수행한다. 이 과정을 통해 두 개 키워드가 모두 출현한 문서 집합 D을 정하고 D에 대해서 두 키워드 각각의 DRE 데이터 위치를 얻을 수 있다. 이어서 랭킹 연산을 수행한다. DRE 데이터는 집합 D에 대해서만 읽으면 되기 때문에 선택적 읽기가 가능하다. 이 때도 디스크 탐색지연 시간을 고려하여 D에 속하지 않은 문서의 DRE 데이터도 함께 읽어야 하는 경우는 존재한다.

(그림 2)에서와 같이 DRE 데이터를 DID와 분리시키고 이에 대한 주소 정보를 두 번째 계층 파일에 추가한 것은 질의 처리시 DRE 데이터가 상당 부분 읽혀 질 필요가 없기 때문이다. 다중 키워드 질의의 경우 동등조인을 통해 선택되는 문서 비율인 선택도(selectivity)가 낮을 수 있기 때문에 DRE 데이터의 경우 전체 데이터 중 읽혀 질 데이터의 비율이 작다. 또한, DRE 데이터는 DID 데이터 보다 그 크기가 7-8 배 이상이기에 전체 DRE를 읽을 경우 디스크 비용이 과도할 수 있다. 따라서 선택적인 읽기를 위해 DRE 데이터를 따로 분리하고 이에 대한 접근 정보를 DID 리스트와 함께 두는 방식을 취한다.

3.2 제안하는 알고리즘

본 절에서는 3.1절에서 설계된 색인 파일을 사용하여 다중 키워드 질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다.

(그림 2)의 색인 파일을 사용하여 5,000만 개 정도의 웹 문서를 색인하여 검색한 결과 과도한 디스크 사용과 질의 처리 시간을 경험했다. 예를 들어 키워드 세 개로 구성된 질의가 있고 키워드의 문서 출현수가 10만 건으로 동일하다고 해보자. 이 경우 문서 식별자와 DRE 데이터 위치를 저장하기 위해 대략 2.4MB 크기의 DID 색인을 가진다. 동등조인 연산 과정에서 이 데이터를 모두 읽어야 하고, DRE 데이터도 랭킹 연산을 위해 읽어 들어야 한다. 경험적으로 DRE 데이터를 읽는데 보통 DID 데이터를 읽어 들이는 비용의 3배 정도가 소요되기에, 전체 10MB에 가까운 디스크 데이터를 읽어 들이는 디스크 비용이 예상된다. 웹 문서 검색에서 이런 정도의 키워드 빈도는 특별한 상황이 아니기 때문에 하나의 디스크에서 처리된다면 수 초가 넘는 디스크 시간이 발생한다. 다수의 사용자 질의를 동시 처리해야 하고 적절한 수준의 질의 응답 시간을 보장하기 위해서는 보다 효과적인 동적 색인 파일 관리 기법이 필요했다.

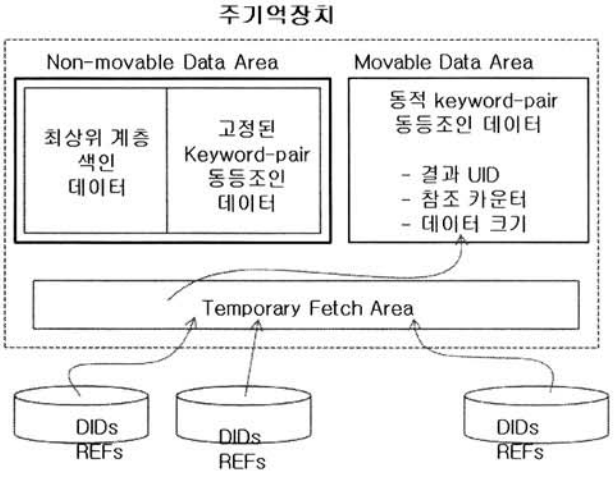
제안하는 동적 색인 관리 기법의 설명을 위해 논문에서 사용하는 개념 및 용어에 대해서 먼저 기술한다. 개발된 기법은 상용 웹 검색 시스템에 적용된 것이기에 사용자의 입력 질의를 저장한 로그 파일을 가지고 있으며 이를 통해 다중 키워드 질의 집합을 수집할 수 있다. 이렇게 수집된 다중 키워드 질의 집합을 S_m이라 하고 이에 속한 임의의 질의 Q를 고려해 본다. 질의 Q는 두 개 이상의 키워드로 구성되며, Q의 키워드 중 서로 다른 두 개를 하나의 쌍으로 하는 키워드쌍 집합 W_q를 생성할 수 있다. 예를 들어 질의 Q $=\{k_1, k_2, k_3\}$일 때, $W_q = \{ k_1k_2, k_1k_3, k_2k_3 \}$의 키워드쌍 집합을 구할 수 있다. 논문에서는 이런 관계를 "W_q made-from Q"라 한다. 그리고, W_q에 속한 단어를 워드쌍(keyword pair)이라 부르고, 하나의 워드쌍 w에 키워드 k가 포함되어 있다면 "w include k"의 관계가 있다고 한다.

이제 이런 정의를 통해 주기억장치에서 관리될 워드쌍을 선택하는 과정을 설명한다. 로그 데이터에서 모든 다중 키워드 질의어를 추출하고 이를 S_m에 저장하며 이때 질의 인기도를 함께 기록한다. 예를 들어 다중 질의어 Q가 로그에 n번 입력되었다면, Q의 인기도는 n이 된다. 이제 S_m에 속한 다중 키워드 중 인기도 상위 K개의 질의를 선택한 후 이를 S라 한다. 그리고, 아래와 같이 S로부터 생성 가능한 모든 워드쌍을 구해 W_a에 저장하며 이 때 저장된 워드쌍의 인기도 역시 함께 계산한다.

```
for every query Q in S
    W ← made-from(Q ). // "W made-from q"의 관계 성립
    for every keyword-pair w in W
        Increase the popularity of w and save it in W_a.
    endfor
    endfor
```

앞에서와 같이 W_a에 속한 각 워드쌍의 인기도를 구하고 이 중에서 시스템에서 정한 N개의 상위 인기도 워드쌍을 선택하여 이를 W_p라 한다. 이렇게 정한 W_p는 다중 키워드 질의를 구성하는 키워드 쌍 중 자주 입력되는 쌍이다. 제안 알고리즘은 W_p에 속한 두 키워드에 대해 동등조인을 수행한 결과를 주기억장치에 저장함으로써 질의 처리 비용을 최소화 시킨다.

(그림 3)은 제안하는 주기억장치 공간의 사용 모습이다. 주기억장치 공간은 NDA(Non-movable Data Area), MDA (Movable Data Area), 그리고 TFA(Temporary Fetch Area)로 나뉜다. NDA에는 (그림 2)에서 설명한 최상위 계층 색인 데이터와 앞서 정의한 인기 워드쌍 집합 W_p 중에서 최상위 인기도 워드쌍의 동등조인 결과가 저장된다. NDA에 저장된 데이터는 시간 흐름에 관계없이 그 내용이 고정된다. 이와는 다르게 MDA에 저장되는 데이터는 시스템에 입력되는 사용자 질의 유형에 따라 저장된 워드쌍 집합이 변한다. MDA에도 워드쌍에 대한 동등조인 결과가 저장되며 동적인 관리를 위한 추가 데이터인 참조 카운터와 결과 DID 크기가 저장되어 있다. TFA에는 실시간으로 읽어 들이는 DID 데이터나 DRE 데이터가 임시 저장되며 이 중 일부는 제안하는 알고리즘에 따라 MDA로 이동한다. TFA의 크기는 동시 처리되는 질의 수에 따라 정하며, TFA로 읽어 들이는 데이터는 운영체제 버퍼를 거치지 않고 디스크에서 직접 읽어 들임으로써 운영체제 버퍼로부터의 메모리 복사



(그림 3) 주기억장치에서의 데이터 관리

비용을 줄인다.

제안하는 색인 파일 동적관리 기법은 (그림 4)의 다중 키워드 질의 처리 알고리즘에 적용된다. (그림 4)의 *Process_Query* 알고리즘은 네 개의 입력 변수를 받는다. 변수 중 Q는 처리할 질의를, W_p는 로그 분석을 통해 미리 정한 인기 워드쌍 집합이다. 또, W_n은 NDA 영역에 저장된 워드쌍 집합을, W_m은 MDA 영역에 저장된 워드쌍 집합을 표시한다. W_m에는 동적 저장에 필요한 데이터들, 즉 동등조인의 결과로 나온 DID 데이터의 MDA 내 참조 횟수나 크기 등도 포함된다. 알고리즘은 동등조인 연산과 랭킹 연산을 수행한 후 랭킹 값에 따라 정렬된 DID 리스트를 반환한다. 반환된 DID 리스트에 따라 검색 결과 화면이 생성될 것이다.

알고리즘 수행 단계는 다음과 같다. 단계 1에서 입력된 질의 Q로부터 워드쌍 집합 W_q를 생성한다. 이어지는 단계 2-5에서 W_q에 속한 워드쌍 중 W_m 혹은 W_n에 있는 것을 선택하여 집합 *Cover*에 저장한다. 단계 6에서는 집합 *Cover*에 속한 워드쌍이 질의 Q를 포괄(cover)하는지 체크한다. 만약 포괄하고 있다면 이는 Q의 동등 조인에 필요한 DID 데이터를 W_n 및 W_m에서 얻을 수 있음을 의미한다. 여기서 워드쌍 집합 *Cover*가 Q를 포괄한다고 함은 아래 두 조건이 모두 만족될 때이다.

- *W_q가 "W_q made-from Q"인 워드쌍 집합이고, 집합 Cover는 W_q의 부분 집합.*
- *모든 키워드 $k \in Q$에 대해서, "w include k"인 워드쌍 w가 Cover에 하나 이상 존재.*

포괄의 의미를 간단히 알아보기 위해 워드쌍 집합 W = {*ab, bd, cd* }, 질의 Q = {*a, b, c, d* }를 가정해 본다. 위의 조건에 의해 W는 Q를 포괄하는 관계이다. 또한 워드쌍 *ab*와 *cd*의 대등조인 결과를 대등조인 함으로써 Q에 대한 문서집합을 얻을 수 있다. 이와 달리 W = {*ab, bd* }, Q = {*a, b, c, d* }일 때는 키워드 *c*에 대한 DID 데이터가 필요하며 이는 디스크로부터 읽어 와야 한다. 따라서 단계 6에서 포괄 관계를 만족한다면 단계 7에서와 같이 디스크를 사용 없이 동등조인을 수행한다. 만약 포괄 관계가 성립하지 않는다면 단계 9-12를 통해 필요한 DID 데이터를 디스크에서 읽어 들인다.

Algorithm Process_Query(Q, W_p, W_n, W_m)
Q: entered query
W_p: keyword pair set made from the query log
W_n: keyword pair set stored in the NDA
W_m: keyword pair set stored in the MDA

Begin
1. $W_q \leftarrow$ ***made_from***(Q). // S_{query}는 q에서 생성되는 워드 집합
2. *Cover* $\leftarrow \emptyset$. // 초기화/
3. for $\forall$ w $\in W_q$ do
4. *Cover* $\leftarrow$ *Cover* $\cup$ {w}, if $w \in W_n$ *or* $w \in W_m$. // 메모리 저장 워드들 수집
5. endfor
6. if covered(*Cover* , Q) then
7. NDA와 MDA에 저장된 데이터 만으로 동등조인 수행; 라인 20으로 이동. // 디스크 사용 없음
8. else
9. for $\forall$ k $\in Q$ do
10. if not covered(*Cover* , {k})
11. 디스크에서 키워드 k의 DID 데이터를 읽어 들임. // TFA 영역 사용
12. endfor
13. NDA, MDA, TFA에 있는 DID 데이터를 사용하여 동등조인 수행.
14. for $\forall$ w $\in W_q$ do
15. if w $\in W_p$ and w $\in W_n$ then
16. 함수 InsertMDA(w, w의 DID 데이터) 호출. // 캐시 관리 알고리즘 적용
17. endfor
18. endif
19. for $\forall$ w $\in W_q$ do
20 if w $\in W_m$ then
21. MDA에 기록된 워드 w의 참조 회수를 하나 증가 시킴.
22. endfor
23. 구한 동등조인 결과를 사용하여 랭킹 연산을 수행. // DRE 데이터 읽혀짐.
24. 랭킹 값 높은 순으로 정렬한 후 결과 DID 리스트를 반환.
End.

(그림 4) 질의 처리 알고리즘 *Process_Query*

단계 14-17에서는 Q로부터 생성되는 워드쌍 중에서 W_m에 추가할 워드쌍이 있는지 체크한다. 이를 위해 W_q의 원소 중에서 W_p 에 속하면서 W_n에 속하지 않은 워드쌍을 찾아 MDA에 추가한다. 이를 위해 단계 16에서 InsertMDA()를 호출한다. 이 함수는 워드쌍의 검색 인기도를 감안하여 MDA 삽입 여부를 결정한다. 인기도는 참조 회수 f와 동등조인 DID 데이터 크기 d에 의해 정해지며 $f/\sqrt{d}$의 값에 비례한다. 새로운 워드쌍 추가 시에 현재 MDA에 있는 워드쌍 중 인기도가 낮은 희생자(victim)를 선택하게 되며, 만약 삽입할 워드쌍보다 인기도가 더 낮은 워드쌍이 존재하지 않을 경우 MDA에 추가되지 않는다. 새로 추가할 워드쌍 w는 MDA에 추가될 때 참조 회수를 1보다 크게 함으로써 새로 삽입된 워드가 곧바로 희생자가 선택되는 것을 피한다. 또

한 시간에 흐름에 따라 참조 빈도가 낮아지는 워드쌍을 찾아 내기 위해 MDA에 속한 워드쌍의 참조 횟수를 하나씩 주기적으로 감소시킨다.

이어서 단계 19-22에서는 질의 Q에 관련된 워드쌍의 참조 회수를 증가시키기 위해 MDA 데이터를 수정한다. 다음 23 단계에서는 이제까지 구한 동등조인 결과를 사용하여 랭킹 연산에 필요한 DRE 데이터를 읽는다. DRE 데이터를 읽은 후에는 랭킹 연산 알고리즘에 따라 문서 관련도를 계산하고 그 값에 따라 정렬하여 결과를 반환한다.

4. 성능 평가

앞 장에서는 대용량 문서의 전문검색(full-text search)에 적합한 색인 파일 계층화 기법과 사용자 질의를 고려하는 색인 데이터의 동적 관리 기법을 기술하였다. 본 장에서는 이런 기법 적용에 따른 디스크 비용 절감 효과를 분석한다. 이를 위해 제안하는 방법을 적용하지 않을 경우의 디스크 비용 C_{naive}와 기법 적용 후의 비용인 C_{enh}를 구하여 이를 서로 비교하는 방식을 취한다.

디스크 읽기 비용 산정 시 어려운 점은 디스크 비용과 읽어야 할 데이터 크기가 서로 정비례하지 않는다는 것이다. 디스크 장치는 주기억장치와 다르게 탐색지연 및 회전지연 시간이 있기 때문에 동일한 크기의 데이터를 읽어 들여도 해당 데이터가 디스크 트랙에 연속해 위치하는가 그렇지 않은 가에 크게 좌우된다. 또한, 최신 디스크 드라이브에서는 디스크 데이터가 트랙에 연속해 존재한 다면 수십 바이트 크기의 데이터를 읽어 들이는 비용이나 수십 킬로 바이트를 읽어 들이는 비용이나 차이가 없다. 이는 빠른 디스크 회전 속도와 디스크 내 버퍼 메모리 사용에 기인한다[18]. 이런 이유로 논문에서는 연속된 32KB 크기의 디스크 데이터를 읽어 들일 때 필요한 비용을 기본 디스크 비용으로 삼는다. 이런 기본 디스크 비용을 DTU(Disk Time Unit)이라 칭하고 이 단위를 사용하여 C_{naive}와 C_{enh}의 크기를 표시한다. 이런 기본 단위 개념을 사용한 비용 산정은 기존 다른 논문에서도 적용된 바 있다[6].

먼저 C_{naive}를 DTU 단위로 계산해 본다. 식 (1)의 $C_{naive}(Q)$는 질의 Q의 처리에 필요한 동등조인 및 랭킹 연산에 드는 디스크 읽기 비용을 나타낸다. 식 (1)은 DID 데이터와 DRE 데이터를 서로 분리하지 않고 서로 인접해 저장한 후 같이 읽어 들이는 것을 가정한다. 식에서 $N_k(Q)$는 질의 Q의 키워드 개수이며 Q를 구성하는 개별 키워드는 k_i로 표시한다. DID(k_i)는 DTU로 계산된 k_i의 DID 데이터 크기를, DRE(k_i)는 DTU로 계산된 k_i의 DRE 데이터 크기를 의미한다. 앞서 언급했듯이 DRE 데이터는 DID 데이터에 비해 7-8배 정도의 크기를 갖는다. DRE 데이터와 같이 큰 데이터의 경우 디스크 트랙에 연속적으로 저장되어 있음으로 해서 실제적인 디스크 비용은 DID 데이터 읽을 때에 비해 대략 3배 정도의 디스크 비용이 사용됨을 알 수 있었다. 이런 점을 고려한 것이 식 (1)의 두 번째 식이다. 아래 식에서 키워드 별 DID 데이터 주소(즉, 최상위 계층 데이터)는 주기억장치에 상주하기에 이 데이터의 접근 비용은 없는 것으로 계산한다.

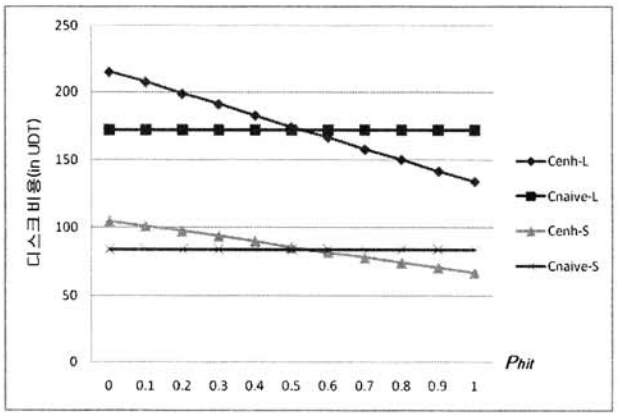
$$C_{naive}(Q) = \sum_{i=1}^{N_k(Q)} [DID(k_i) + DRE(k_i)] = 4 \times \sum_{i=1}^{N_k(Q)} DID(k_i) \quad (1)$$

아래 식 (2)는 제안한 계층적 색인 구조에 동적 색인 알고리즘 적용에 따른 디스크 비용인 C_{enh}을 보인다. 식 (1)에서와 같이 키워드 별 DID 데이터 주소는 주기억장치에 있기 때문에 비용에서 제외한다. 식 (2)에서 P_i는 키워드 k_i의 DID 데이터를 MDA 혹은 NDA에서 발견할 확률을 표시하고, α_i는 키워드 k_i가 MDA 혹은 NDA에서 발견될 때의 비용을 나타낸다. 주로 메모리 복사 비용이므로 실제 운용에 있어 이 값은 DTU의 5%를 넘지 않는다.

$$C_{enh}(Q) = \sum_{..}^{N_k(Q)} (\alpha_i \times P_i) + 2 \times \sum_{.}^{N_k(Q)} (DID(k_i) \times (1 - P_i)) + \sum_{i=1}^{N_k(Q)} \min [DRE(k_i) \times SEL(k_i), DRE(k_i)] \quad (2)$$

식 (2)의 두 번째 항에서 DID 데이터에 2를 곱한 것은 제안된 방식에서는 DID 데이터에 DRE 데이터에 대한 주소값이 저장되고 그 크기가 문서식별자 데이터 크기와 같기 때문에 이를 고려한 것이다. 또한 식 (2)에서 항목 $DRE(k_i) \times SEL(k_i)$은 선택적인 DRE 데이터 읽기를 했을 때의 비용이다. 여기서, $SEL(k_i)$는 질의 Q의 키워드 k_i 동등조인의 선택도(selectivity), 즉 전체 UID 개수에 대한 조인 결과의 비율을 나타낸다. 식 (2)의 비용은 워드쌍의 동등조인 결과를 MDA나 NDA에서 발견할 확률과 질의 선택도에 의존하며, 이 값들은 개별 질의의 특성에 크게 좌우된다. 본 장에서는 식 (1)과 식 (2)에 기반하여 제안하는 기법의 성능 향상을 보인다.

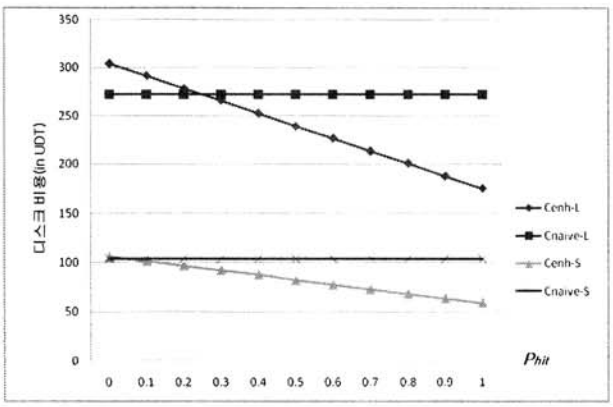
(그림 5)는 키워드의 DID 데이터를 MDA나 NDA의 주기억장치에서 발견할 확률인 P_{hit}의 변화에 따른 성능 향상을 보인다. 그래프에서 C_{enh}는 제안한 방식이고, C_{naive}는 기법 적용을 하지 않았을 때의 디스크 비용이다. 질의를 구성하는 각 키워드가 MDA나 NDA에 존재할 확률은 서로 다르



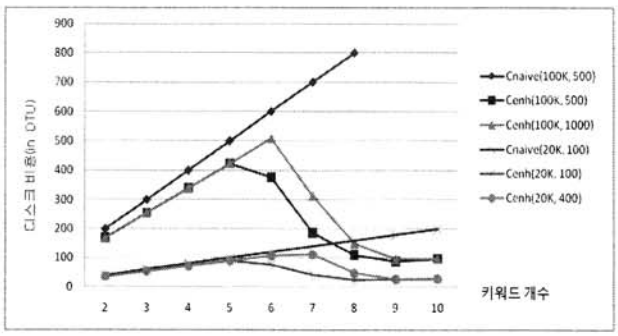
(그림 5) 키워드 3개를 갖는 질의에 대한 성능 향상 예

지만 그림에서는 모두 P_{hit}의 값으로 같다고 가정한다. 그래프는 세 개의 키워드로 이루어진 질의를 처리할 때의 평균 비용이며, 그래프 이름에서 뒤에 '-S'가 붙은 경우는 질의 키워드의 문서 출현 횟수가 각각 10K, 20K, 50K로 비교적 작은 출현 회수를 가질 때의 비용을 나타낸다. 반면 '-L'이 붙은 경우는 질의 키워드의 문서 출현 횟수가 20K, 50K, 100K로 상대적으로 큰 경우이다. 키워드 동등조인 결과 문서 수는 500개로 두었으며, 식 (1)의 α_i 값은 모든 키워드에 대해 UDT의 5%로 두었다. 그림에서 알 수 있듯이 데이터가 큰 경우나('-L'의 경우) 작은 경우 모두 P_{hit}의 값이 0.5를 넘는 범위에서 5%에서 23%까지의 성능 향상을 보인다. 입력 질의 특성이나 사용 가능한 주기억장치의 크기에 따라 차이가 있지만 2GB 주기억장치를 사용하여 구현하였을 때 MDA나 NDA에서의 DID 데이터의 평균 히트율은 90%에 가까운 값을 보였다. 이런 점을 고려한다면 P_{hit} 값은 일반적으로 0.5를 상회하는 값으로 볼 수 있으며, 제안하는 기법을 통해 상당한 성능 향상을 기대할 수 있다. 특히 MDA나 NDA에서 발견되지 않는 경우는 문서 출현 빈도가 다소 낮은 키워드인 경우가 많다. 따라서 DID 데이터를 읽어 들이는 비용이 상대적으로 작기 때문에 실제 얻을 수 있는 성능 향상 분은 (그림 5)에서 보이는 것에 비해 크다고 할 수 있다.

(그림 6)은 (그림 5)와 유사한 환경에서 질의를 구성하는 키워드 개수를 4로 가정했을 때의 성능 그래프이다. 이름에 '-S'가 붙은 경우는 키워드의 문서 출현 수가 10K, 20K, 20K, 50K인 경우이고, '-L'의 경우는 20K, 50K, 100K, 100K로 상대적으로 자주 출현하는 키워드를 포함한 경우이다. 일반적으로 키워드 개수가 많아 질수록 동등조인의 결과 문서 수가 감소함을 고려해서 검색 결과 문서 수는 400으로 했다. (그림 6)에서 보이듯이 (그림 5)에서 보다 작은 P_{hit} 값에서도 성능 향상 효과가 뚜렷하다. 이로 인해 (그림 6)에서는 성능 향상 효과가 최대 55%로 앞의 경우에 비해 보다 좋은 성능 향상 효과가 있었다. 웹 검색의 경우 영문 검색의 경우는 키워드 개수가 3을 넘지 않는 경우가 많지만 한글 검색의 경우는 영문 검색에 비해 하나의 질의에 사용되는 키워드 수가 많을 수 있다[10]. 이는 한글의 경우 형태소 분석을 통해 복합 명사를 짧은 여러 개의 명사로 분석하여



(그림 6) 키워드 4개를 갖는 질의에 대한 성능 향상 예



(그림 7) 키워드 개수 변화에 따른 성능 향상 예

검색해야 하기 때문이다. 따라서 키워드 개수 증가에 따른 성능 향상 효과는 매우 유용한 특성이라 할 수 있다.

(그림 7)은 질의를 구성하는 키워드 개수가 증가할 때의 성능 비교 결과이다. 그래프 $C_{naive}(X, Y)$와 $C_{enh}(X, Y)$에서 X는 키워드의 문서 출현수를 나타내고, Y는 동등조인 결과로 나온 문서 수를 나타낸다. 예로 C_{enh}(20K, 400)이라고 하면 질의에 사용된 키워드의 문서 출현수는 20K이고 두 개 키워드에 대한 동등조인 결과 문서 수는 400을 의미한다. 이 때 키워드 수를 증가시키면 동등조인 결과 문서 수도 감소해야 하는데, (그림 7)에서는 하나의 키워드 추가 시 마다 20%의 동등조인 결과 문서 수 감소를 가정한다. 또한 P_{hit}의 값은 0.85로 하였고, 앞의 경우와 같이 각 키워드의 문서 출현수는 같게 두었다. 그림에서 알 수 있듯이 문서 출현 수가 큰 경우에 보다 많은 성능 향상을 일어남을 알 수 있고, 키워드 수가 6개 이상부터는 매우 급격한 성능 향상이 일어남을 알 수 있다. 사용자의 실제 검색 형태를 살펴보면 키워드 수가 5개를 넘는 경우는 많지 않다. 하지만 사용자 검색 창에서의 키워드 붙여 넣기 실수나 웹 문서 수집 로봇 등의 영향으로 일반적인 경우를 벗어나 매우 많은 키워드가 하나의 질의에 입력되는 경우들이 있다. 이 경우 검색 시스템에 매우 큰 작업 부하를 주며 이로 인해 정상적인 질의의 처리에 영향을 주기도 한다. 하지만 제안된 방식에서는 이런 경우 매우 빠른 속도로 시스템 부하가 감소되기에 전체 시스템 성능 유지에 유리하다고 할 수 있다.

5. 결 론

논문에서는 대용량의 문서를 색인 한 전문검색 서비스에 맞는 색인 파일 구조 및 주기억장치를 사용한 색인 파일의 동적 관리 기법에 대해서 연구했다. 제안된 색인 파일은 다중 키워드 질의를 처리하기 위해 수행되는 동등조인 연산과 랭킹 연산에 적합하도록 계층화 구조를 취한다. 계층 구조는 키워드 별로 문서 식별자 리스트를 접근하기 위한 최상위 색인 계층을 가지며 이 정보는 주기억장치에 저장된다. 두 번째 계층 색인 파일에서는 키워드 별 문서식별자 리스트와 함께 키워드의 문서 내 색인 정보가 저장된 디스크 위치가 함께 저장된다. 이 곳에 저장된 디스크 주소 정보를 이용하여 랭킹 연산에 필요한 데이터를 선택적으로 읽을 수 있다.

제안하는 방식은 디스크 비용을 줄이기 위해 두 개 키워드 간 동등조인 결과를 주기억장치에 동적으로 저장할 수 있도록 사용자 질의를 분석하여 입력 빈도가 높은 키워드 쌍을 정하고 이들 중에 빈도가 아주 높은 키워드 쌍에 대해서 동등조인 결과를 주기억장치에 정적으로 저장한다. 그리고 나머지 키워드 쌍들은 주기억장치에서 동적으로 관리된다. 이런 방식을 통해 다중 키워드 질의 처리에 소요되는 디스크 비용을 크게 줄일 수 있었다. 제안된 방식에 의하면 주기억장치에 저장된 키워드 쌍이 검색되는 비율이 50% 이상인 경우 성능 향상을 얻을 수 있다. 실제 시스템의 경우 그 값이 90%에 근접하기 때문에 제안한 방식을 통해 성능 향상 효과를 얻을 수 있었다. 특히 질의의 키워드 개수가 증가할 때 제안하는 방식을 통해 매우 큰 디스크 비용 감소 효과를 얻을 수 있다. 이런 특징은 한국어와 같이 형태소 분석을 통해 키워드 개수가 쉽게 증가할 수 있는 상황에서 유용한 특징이라 할 수 있다.

참 고 문 헌

[1] 이주남, Google과 함께 떠오르는 검색엔진, 소프트웨어진흥원 시장 이슈 보고서, 2004.

[2] Steve Lawrence, C. Lee Giles, and Kurt Bollacker, "Digital Libraries and Autonomous Citation Indexing," IEEE Computer, Vol.32, No.6, pp.67-71, 1999.

[3] Arvind Arasu, et al., "Searching the Web," ACM Trans. on Internet Technology, Vol.1, No.1, pp.2-43, August, 2001.

[4] Search Engine Report, Http://www.searchenginewatch.com, 2010.

[5] Sergey Brin, Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Computer Networks and ISDN Systems, Vol.30, Issue 1-7, pp.107-117, 1998.

[6] 임성채, "계층적 캐시 기법을 이용한 대용량 웹 검색 질의 처리 시스템의 구현", 정보과학회논문지 : 컴퓨팅의 실제 및 레터, Vol.14, No.7, pp.669-679, 2008.

[7] Maxim Lifantsev and Tzi-cker Chiueh, "I/O-Conscious Data Preparation for Large-Scale Web Search Engines," In Proc. the VLDB Conf., Hong Kong, 2002.

[8] Sergey Melnik, Sriram Raghavan, Beverly Yang, and Hector Garcia-Molina. "Building a Distributed Full-text Index for the Web," In Proc. of the 10th International World Wide Web Conference, pp.396-406, 2001.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System," 19th ACM Symposium on Operating Systems Principles, October, 2003.

[10] Soyeon Park, Joon Ho Lee, and Hee Jin Bae, "End user searching : A Web log analysis of NAVER, a Korean Web search engine," Vol.27, No.2, pp.203-221, 2005.

[11] Maxim Lifantsev and Tzicker Chiueh, "Implementation of a Modern Web Search Engine Cluster," In Proc. of the USENIX Annual Technical Conference, Texas, 2003.

[12] BoostingCraig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz, "Analysis of a very large web search engine query log," ACM SIGIR Forum, Vol.33(1), pp.6-12, 1999.

[13] Tiziano Fagni, Raffaele Perego, Fabrizio Silvestri, and Salvatore Orlando, "Boosting the performance of Web Search Engines: Caching and Prefetching Query Results by Exploiting Historical usage Data," ACM Trans. on Information Systems, Vol.24(1), pp.51-78, 2006.

[14] Ronny Lempel and Shlomo Moran, "Predictive Caching and Prefetching of Query Results in Search Engines," In Proc. of the 12th International Conf. on World Wide Web, pp.19-28, New York, 2003.

[15] Hao Yan, Shuai Ding, and Torsten Suel, "Inverted Index Compression and Query Processing with Optimized Document Ordering," In Proc. of the WWW Conference, pp.401-410, 2009.

[16] Vo Ngoc Anh and Alistair Moffat, "Inverted Index Compression Using Word-Aligned Binary Codes," Information Retrieval, Vol.8, No.1, pp.151-166, 2005.

[17] Sung-Ryul Kim, Inbok Lee, and Kunsoo Park, "A fast algorithm for the generalized k-keyword proximity problem given keyword offsets," Information Processing Letters, Vol.91, No.3, pp.115-120, 2004.

[18] C. Ruemmler and J. Wikes, "An Introduction to Disk Modeling," IEEE Computer, Vol.17, No.3, pp.17-28, 1994.

임 성 채

e-mail : sclim@dongduk.ac.kr

1992년 서울대학교 컴퓨터공학과(학사)

1994년 KAIST 전산학과(석사)

2002년 KAIST 전산학과(박사)

2000년~2005년 코리아와이즈넛 수석연구원 및 이사

2005년~현 재 동덕여자대학교 컴퓨터학과 조교수

관심분야 : 이동 객체 색인기법, 데이터마이닝, Web IR