J-tree: 사용자의 검색패턴을 이용한 대용량 데이타를 위한 효율적인 색인

(J-Tree: An Efficient Index using User Searching Patterns for Large Scale Data)

장 수 민 † 서 광 석 † 유 재 수 ††
(Sumin Jang) (Kwangseok Seo) (Jaesoo Yoo)

요 약 최근에 휴대용 단말기들의 발전으로, 대용량 데이타에 대한 다양한 검색 서비스들이 휴대용 단말기에 제공되고 있다. 정보 검색을 위한 대부분 응용프로그램들은 대용량 데이타를 검색하기 위하여 B-tree나 R-tree와 같은 색인을 사용한다. 그러나 전체 데이타의 매우 적은 부분이 사용자에 의하여 접근된다. 또한, 각 데이타에 대한 접근 빈도수들은 다양하다. 그러나 B-tree나 R-tree와 같은 색인들은 편향적 접근 패턴의 특성을 고려하지 않는다. 그리고 캐쉬는 빠른 접근을 위해서 반복적으로 접근되는 데이타를 메모리에 저장한다. 그러나 캐쉬에서 사용하는 메모리의 크기는 제한적이다. 본 논문에서는 사용자의 검색패턴들을 고려한 디스크 기반의 새로운 색인구조, J-tree를 제안한다. 제안된 색인은 모든 데이타에 대한 일정한 검색속도를 보장하는 균형트리이다. 그리고 자주 접근된 데이타에 대해서는 빠른 검색속도를 제공한다. 성능평가는 다양한 실험환경에서 제안된 색인의 효율성을 보여준다.

키워드 : 검색 색인, 편향 접근 패턴, 사용자 검색 패턴

Abstract In recent years, with the development of portable terminals, various searching services on large data have been provided in portable terminals. In order to search large data, most applications for information retrieval use indexes such as B-trees or R-trees. However, only a small portion of the data set is accessed by users, and the access frequencies of each data are not uniform. The existing indexes such as B-trees or R-trees do not consider the properties of the skewed access patterns. And a cache stores the frequently accessed data for fast access in memory. But the size of memory used in the cache is restricted. In this paper, we propose a new index based on disk, called J-tree, which considers user's search patterns. The proposed index is a balanced tree which guarantees uniform searching time on all data. It also supports fast searching time on the frequently accessed data. Our experiments show the effectiveness of our proposed index under various settings.

Key words : Search Index, Skewed Access Patterns, User Searching Patterns

· 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임

† 학생회원 : 충북대학교 정보통신공학과
jsm@cbnu.ac.kr
proudseo@naver.com
†† 종신회원 : 충북대학교 정보통신공학과 교수
yjs@cbnu.ac.kr

논문접수 : 2008년 10월 13일
심사완료 : 2008년 12월 11일

정보과학회논문지: 데이타베이스 제36권 제1호(2009.2)

1. 서 론

최근에 휴대용 단말기를 이용하는 개인사용자가 급증하고 있는 실정이다. 이러한 휴대용단말기를 이용한 대용량의 데이타를 검색하는 휴대용 내비게이션과 같은 다양한 응용프로그램이 사용되고 있다[1]. 이러한 응용프로그램을 이용하는 사용자는 많은 데이타에 대한 빠른 검색서비스를 요구하고 있다. 그러나 기존의 검색방법은 생성된 데이타의 속성 값을 기반으로 색인을 생성하고 검색서비스를 제공하고 있다. 그러나 검색서비스를 제공하는 응용프로그램은 전체 데이타 중에 사용자가 실제 검색하여 참조하는 데이타는 약 20%~30%정도로 전체 데이타의 일부분에 해당한다. 이처럼 검색의 대상

이 전체 데이타의 일부분에 편중되어 있다. 그래서 본 논문에서는 이러한 사용자의 검색패턴을 검색을 위한 색인에 반영하여 보다 빠른 검색서비스를 제공하는 색인구조를 제안한다. 제안하는 색인구조는 사용자의 검색패턴을 분석하고 이를 색인에 반영한 새로운 색인이다. 이러한 사용자의 검색패턴을 반영한 색인은 검색을 위해 하드디스크에 있는 데이타를 메모리에 가져오기 위한 IO작업의 일부를 생략함으로써 보다 빠른 검색을 제공한다. 본 논문에서 제안하는 색인은 데이타에 대한 삽입과 삭제가 실시간으로 변경되지 않고 특정기간이 지난 이후에 일괄적으로 반영되는 검색서비스에 초점을 맞춘다.

본 논문의 구성은 다음과 같다. 2장에서 관련된 연구를 소개하고 3장에서는 본 논문에서 제안하는 J-Tree기법 및 특성을 기술한다. 마지막으로 4장에서 다양한 실험환경에서 제안하는 기법과 B^+-Tree의 성능평가를 통하여 제안하는 J-Tree 색인의 우수성을 입증한다.

2. 관련연구

최근에 사용자의 검색패턴을 이용한 검색서비스가 웹 검색서비스에서도 사용되고 있다[2,3]. 사용자가 자주 사용하는 검색 키워드를 이용하여 보다 빠른 서비스를 제공하는 형태로 이루어지고 있다. 또한 콘텐츠에 대한 접근 횟수를 반영하여 개인의 성향을 맞춘 형태의 웹 서비스를 제공하거나 사용자가 자주 사용하는 검색어를 캐쉬형태로 유지하여 보다 빠른 검색을 제공하고 있다. 그러나 이러한 방식에는 전체 데이타에 대한 평균적 검색속도의 저하 및 메모리 용량의 한계등 문제점이 있다.

2.1 Alphabetic Huffman Tree

사용자가 자주 참조하는 데이타를 색인의 상위부분에 가깝게 설정하는 편향트리의 대표적인 방법으로는 Alphabetic Huffman Tree(AH-Tree)가 있다. AH-Tree는 멀티채널 환경에서의 브로드 캐스트에 많이 사용되는 색인이다[4]. AH-Tree는 Hu-tucker 알고리즘을 사용하여 생성된다. 이 알고리즘은 먼저 데이타를 접근 빈도에 따라 정렬한 후, 접근 빈도가 낮은 데이타부터 결합하여 트리를 생성하는 상향식 트리 구성 방법을 사용한다. 따라서 AH-Tree에서 트리의 깊이는 각 데이타에 대한 접근 빈도를 나타내게 된다. 즉, 접근 빈도가 높은 데이타는 트리의 상위 부분에 위치하고 접근 빈도가 낮은 데이타는 트리의 하위 부분에 위치하게 됨으로써 접근 빈도가 높은 데이타는 접근 빈도가 낮은 데이타에 비해 상대적으로 작은 탐색시간을 가지게 된다. 그림 1은 1차원 데이타에 대한 AH-Tree의 생성 예이다. (I, 7)보다 상대적으로 접근 빈도가 높은 (E, 77)가 상위 노드에 위치함을 볼 수 있다.

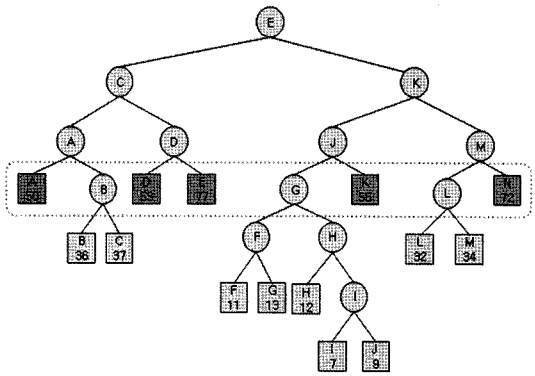

그림 1 Alphabetic Huffman Tree의 구성 예제

AH-Tree는 접근 빈도가 높은 데이타에 대해서 짧은 탐색 시간을 제공하지만 색인 트리를 생성 및 유지하는데 비용이 매우 높다. 그러므로 AH-Tree는 사용자의 검색패턴에서 보이는 편향성은 반영할 수는 있지만 일부 데이타는 탐색경로가 증가하게 되어서 짧은 탐색시간을 보장할 수는 없는 단점이 있다.

2.2 캐쉬

캐쉬는 보조기억장치로서, 특히 디스크의 입출력 효율을 높이기 위해 주기억 장치의 한 영역에 최근에 사용된 디스크 블록의 내용을 기억하는 버퍼영역으로 할당한 것을 말하는데, 캐쉬에 있는 디스크 블록은 디스크 액세스 없이 바로 이용할 수 있으므로 효율이 높아지게 된다[5,6]. 이것은 디스크로부터 한번 읽어 들인 정보를 메모리에 상당시간 보관함으로써 처음 로딩하는 과정과는 달리 중복적인 접근 시에는 버퍼 메모리에 저장되어 있는 데이타를 읽기 때문에 매우 빠른 검색을 제공한다. 이러한 캐쉬 정책의 한계는 메모리 크기에 한정된다. 그래서 대용량 데이타에 대한 검색서비스에 캐쉬을 적용하는 것은 적합하지 않다. 본 논문은 이러한 문제점을 해결하기 위해서 메모리를 이용하는 방법이 아닌 디스크 기반의 색인을 제안한다.

2.3 편향성트리와 균형트리의 비교

편향성 트리는 사용자의 접근 빈도가 높은 데이타에 대해서 짧은 탐색시간을 제공하지만 트리를 생성 및 유지하는데 고비용이 필요하다. 또한 전체 데이타에 대한 평균 검색속도가 균형트리보다 느린 단점을 갖고 있다. 그러나 균형 트리는 사용자의 접근 빈도를 색인에 반영하지 않았기 때문에 빈도가 높은 데이타나 그렇지 않는 데이타나 같은 탐색시간을 제공한다. 그래서 균형트리는 접근 빈도가 높은 데이타에 대해서 보다 빠른 검색을 제공하지 못하는 단점을 갖고 있다. 그림 2는 편향성트리와 균형트리의 비교한 것이다. 편향성 트리는 접근성이 높은 단말노드가 루트로부터 가까운 위치에 설정되

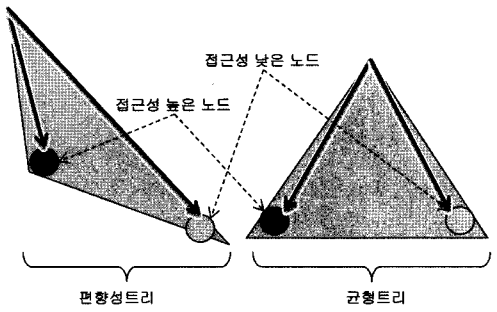

그림 2 편향성트리와 균형트리의 비교

어 있다. 그러나 접근성이 낮은 단말노드는 루트로부터 매우 멀리 떨어져 있다. 또한 균형트리는 모든 데이타에 대하여 균등한 검색속도를 제공하지만 접근 빈도가 높은 데이타에 대하여 빠른 검색속도를 제공하지 못하는 단점을 갖고 있다. 그래서 본 논문에서 제안하는 색인은 접근 빈도가 높은 데이타에 대한 빠른 검색속도를 제공하면서 접근 빈도가 낮은 데이타에 대해서는 균등한 검색속도를 제공하는 균형트리로 설계한다.

3. J-Tree

대표적인 검색서비스는 웹정보검색서비스나 휴대용 내비게이션과 같은 정보검색서비스이다. 이러한 검색서비스는 검색의 대상이 되는 데이타가 대용량이면서 정보를 빠르게 검색할 수 있도록 색인을 제공한다. 최근 정보검색서비스는 개인적 성향을 색인에 반영하는 사례가 증가되고 있다[7,8]. 정보검색서비스의 검색과정은 사용자가 정보를 찾기 위하여 질의어를 보내면 색인을 통하여 그 질의어에 해당되는 결과를 제시한다. 이때 사용자가 정보검색을 위해 사용하는 질의어의 대부분은 같은 질의어를 반복하여 사용하는 경우가 많다. 본 논문에서는 이러한 사용자의 검색패턴을 색인에 반영하여 보다 빠른 검색속도를 제공하는 J-Tree라는 새로운 색인을 제안한다. 그림 3은 이러한 사용자 검색 패턴을 적용한 J-Tree의 개념도를 보여준다.

제안하는 J-Tree는 B^+-Tree와 유사한 균형트리이다. J-Tree는 균형트리의 장점인 모든 데이타에 대한 평균적인 검색속도를 제공할 뿐만 아니라 사용자가 많이 접근하는 데이타는 보다 빠르게 검색되도록 두 가지 특성을 모두 고려하였다. 이러한 사용자 검색 패턴을 색인에 반영하기 위해서는 사용자의 검색 패턴에 대한 정보를 유지해야 한다. 사용자의 검색패턴의 정보는 별도의 과정을 통하여 유지되는 것이 아니라 검색과정에서 유지한다. 사용자의 질의를 처리하는 과정은 루트로부터 내부 노드의 키 값과 비교를 통하여 최종결과를 접근하게

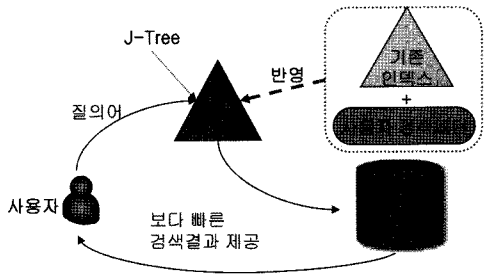

그림 3 사용자 검색 패턴이 적용된 J-Tree의 개념도

되는데 이때, 검색결과에 해당하는 각각의 데이타에 접근 빈도수를 유지하고 이러한 접근 빈도수를 내부노드의 링크들에 대한 접근 횟수로 반영하여 유지한다.

J-Tree는 점프경로(Jump-Path)라는 것을 생성하여 사용자의 검색패턴을 색인에 반영한다. 그림 4는 J-Tree를 생성하기 위해 필요한 점프경로의 개념도이다. 그림 4에서 각 링크는 검색을 위해 접근된 횟수가 유지되고 있다. 이처럼 각 내부노드는 다양한 접근 횟수를 갖는 여러 개의 링크들로 구성된다. 본 논문에서는 이러한 링크들 중에 상대적으로 접근 빈도가 높은 링크가 있는데 이러한 링크들의 연속된 경로를 점프경로(Jump-Path)로 정의한다. 이러한 점프경로는 시작링크, 내부링크들과 끝노드로 연결되는 경로로 구성된다. 점프경로가 의미하는 것은 사용자가 특정 데이타를 검색하기 위해 반복적으로 사용한 질의어에 대하여 빈번하게 참조된 링크들의 경로이다. 이러한 점프경로는 전체 색인에 다수 개가 발생된다.

J-Tree는 점프경로의 시작링크에 점프경로의 끝노드로 분기하기 위한 조건과 분기할 끝노드의 위치를 유지한다. 검색과정에서 점프경로의 시작링크에 접근될 때 끝노드로 분기되는 조건을 미리 비교하여 만족한다면 점프경로의 중간링크들을 비교하지 않고 바로 끝노드로 점프한다. 이때 중간 링크들을 메모리로 가져오는 IO작

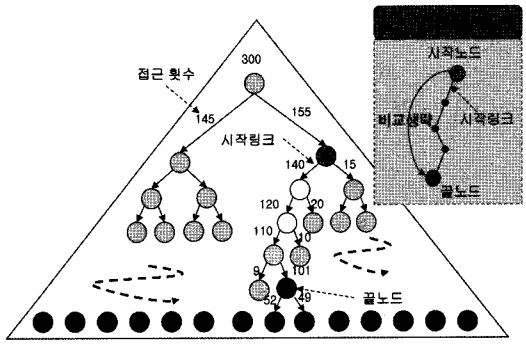

그림 4 점프경로(Jump-Path)의 개념도

업이 생략됨으로써 보다 빠른 검색속도를 제공한다. 만약 끝 노드로 분기하기 위한 조건을 만족하지 못 할 경우 기존의 검색과정을 통하여 결과에 접근한다. 점프경로를 이용하여 분기되는 조건이 만족되지 않을 때, 메모리상에서 끝노드로 분기되는 조건을 비교하는 과정이 있지만 아주 적은 비용만이 소요되기 때문에 일반적인 검색과 거의 같은 성능을 보여준다. 이처럼 점프경로를 이용한 검색은 검색과정에 발생하는 내부노드의 비교검색을 위한 IO작업을 생략함으로써 보다 빠른 검색속도를 제공한다. J-Tree는 검색과정에서 실시간으로 발생하는 각각의 데이타에 대한 접근 빈도수를 단말노드에 반영하지만 J-Tree의 구조에 바로 반영하지 않는다. 특정기간이 지난 이후에 데이타에 대한 삽입, 삭제 및 수정으로 일괄적으로 검색데이타를 변경하는 시점과 사용자의 검색패턴을 재 반영하기 위하여 특정한 주기인 M 기간을 설정하여 J-Tree를 재구축한다. 여기서 특정한 주기 M은 검색서비스의 대상이 되는 데이타의 특성에 맞추어 설정한다.

그림 5는 J-Tree의 구조 및 구축과정을 보여준다. J-Tree는 총 접근 빈도수 Total_RCNT와 전체 색인을 구성하는 링크들의 평균 접근 빈도수로 AVG_RCNT를 유지한다. 점프경로를 생성하는 과정에서 접근 빈도가 상대적으로 높은 링크를 선택하기 위하여 내부노드를 구성하는 링크들의 접근 빈도수 RCNT을 유지한다. 각 링크의 접근 빈도수 RCNT는 링크의 조건에 만족하는 단말노드의 데이타에 대한 빈도수 DRCNT을 합산한 빈도수로 설정된다. 또한 점프경로를 결정하기 위하여 링크들의 평균 접근 빈도수 AVG_RCNT보다 높은 AVG_RCNT(1+k)의 빈도수를 설정하여 접근 빈도가 높은 링크를 선택한다. 이때 k의 값은 상수값으로 AVG_RCNT(1+k)의 값이 TOTAL_RCNT 값보다 적도록 설정하여 사용한다. 점프경로는 AVG_RCNT(1+k)의 빈도수보다 높은 빈도수를 갖는 연속된 링크들을 선택하여 생성한다. J-Tree의 구축과정은 단말노드에 유지하고 있는 빈도수 DRCNT을 내부노드를 구축하는 과정에서 반영하여 B^+-Tree와 비슷한 임시트리를 구축한다. 그리고 최종적인 J-Tree는 그 임시트리에서 점프경로를 선택하여 점프하기 위한 조건으로 JCond 조건과 그 조건을 만족하면 분기될 연결링크인 JLink 링크를 추가하여 완성한다.

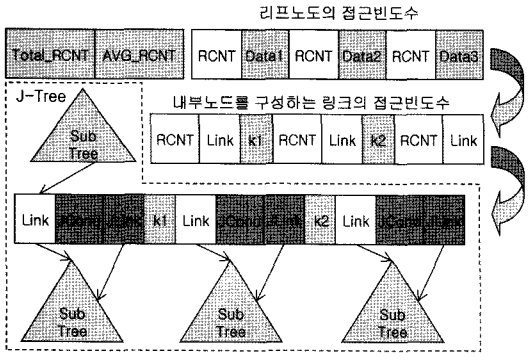

그림 5 J-Tree의 구조 및 구축과정

특히, J-Tree를 재구축하는 시점에서 새로이 추가되는 데이타의 접근 빈도수는 AVG_RCNT의 값으로 설정하여 반영된다. 또한 일정기간동안 유지되는 전체적인 단말노드의 빈도수는 빈도수를 저장하기 위한 변수의 최대값을 넘지 않도록 전체적으로 감소하는 방법으로 유지한다.

4. 성능평가

본 논문에서 제안한 J-Tree를 성능평가하기 위하여 균형트리이면서 데이타 검색에 범용적으로 사용하는 B^+-Tree와 비교 및 분석한다. 실험은 LINUX 운영체계, 펜티엄IV 2.0GHz, 메인메모리 1G를 갖는 시스템에서 실시하였다. 성능평가를 위하여 전체 데이타에 대한 접근 횟수를 미리 설정하기 위하여 십만 번의 검색을 실행하였다. 그리고 점프경로를 생성하기 위하여 AVG_RCNT(1+k)을 설정하기 위하여 k의 상수값을 0.3로 설정하여 점프경로를 결정하였다. 표 1은 성능평가를 위한 설정기호들을 보여준다. S=(a%, b%)는 사용자가 전체 데이타의 특정 영역에 대한 편중된 검색을 설정하기 위한 파라미터이다. S=(90, 10)가 의미하는 것은 전체 검색 횟수의 90%가 전체 데이타의 10%에 해당하는 영역에 질의되는 것을 의미한다.

그림 6은 전체 레코드의 수가 만 개에서 십만 개까지 변화하는 환경에서 질의처리의 속도를 측정한 것이다. 일반 검색서비스에서 대용량 데이타를 B^+-Tree로 처리할 경우 보통 트리의 높이가 약 3레벨에서 5레벨정도로 사용되기 때문에 성능평가를 위하여 트리의 높이가 3레벨에서 5레벨정도가 설정되도록 한 노드에 최대 저장되는 키의 개수를 조정하였다.

그리고 질의에 대한 편중도는 S=(50, 30)으로 전체 검색횟수의 50%가 전체 데이타 영역의 30%에 편중되는 조건에서 측정하였다. 본 논문에서 제안한 J-Tree가

표 1 성능평가를 위한 설정기호

기호	설명
|R|	전체 레코드의 수
S=(a, b)	데이타 검색할 때 전체 검색 횟수의 a 퍼센트가 전체 데이타 영역의 b 퍼센트 범위에서 검색하는 조건
|Q|	전체 질의의 수

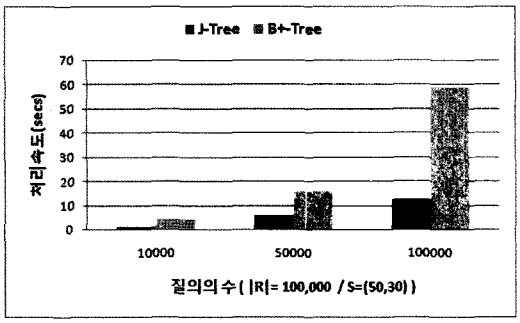

그림 6 질의의 수 변화에 따른 처리속도

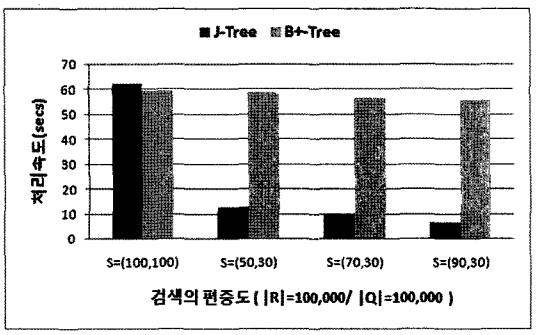

그림 7 검색의 편중도 변화에 따른 처리속도

B^+-Tree보다 레코드의 수가 증가함에 따라 처리속도가 현격히 차이가 나는 것을 보여준다. 이와 같은 결과는 중복되는 질의어에 대한 처리에서 J-Tree가 내부노드의 비교과정을 단축함으로써 처리속도가 빨라진 것이다. 레코드 수의 증가는 색인의 높이를 증가시킴으로써 점프경로에서 중간노드들의 키 비교를 생략하는 정도가 증감함에 따라 보다 빠른 검색속도를 제공한다.

그림 7은 검색의 편중도 변화에 따른 처리속도를 B^+-Tree와 비교하였다. 검색의 편중도가 전혀 없는 S=(100,100)에서는 B^+-Tree가 다소 처리속도가 빠르다. 그러나 검색의 편중도가 늘어남에 따라 제안하는 J-Tree의 검색 속도가 현격히 빠른 것을 보여준다. 그러나 B^+-Tree는 검색의 편중도가 증가하여도 검색속도에는 별 영향이 없음을 보여준다. 이는 전체데이타의 일부분에 편중되는 정보검색서비스에서 J-Tree가 우수함을 증명한다.

5. 결 론

본 논문에서는 일반적인 검색에서 전체 데이타의 일부분에 집중되는 검색을 보다 빠르게 제공하기 위하여 사용자의 검색패턴을 이용한 새로운 색인구조 J-Tree를 제안하였다. J-Tree는 일반적으로 사용하는 B^+-Tree나 R-Tree와 같은 색인보다 점프경로를 이용하여 내부의 검색 비교를 생략함으로써 보다 빠른 검색속도를 제공하였다. 향후연구방향은 본 논문에서 제안하는 J-Tree가 검색서비스에 국한되어있지만 데이타의 삽입과 삭제에 대해서도 실시간으로 이를 색인에 반영하는 구조로 변경할 필요가 있다. 또한 실 검색데이타를 적용하여 보다 세밀한 성능평가를 통하여 보다 향상된 색인을 구현하고자 한다.

참 고 문 헌

[1] Narayanan Shivakumar, Suresh Venkatasubramanian, "Energy Efficient indexing for Information Dissemination In Wireless Systems," in ACM, Journal of Wireless and Nomadic Application, 1996.

[2] Ryen W. White, Dan Morris, "Investigating the querying and browsing behavior of advanced search engine users," Proc, ACM SIGIR, July, 2007.

[3] Yabo Xu, Ke Wang, Benyu Zhang, Zheng Chen, "Privacy-enhancing personalized web search," Proceedings of the 16th international conference on World Wide Web, May, 2007.

[4] S. Lo and A. Chen, "Optimal Index and Data Allocation in Multiple Broadcast Channels," In proceedings. 16th international conference on Data Engineering, 2000.

[5] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide Area Web Cache Sharing Protocol," Proc. ACM SIGCOMM, pp. 254-265, 1998.

[6] K. Wu and P. Yu, "Latency-Sensitive Hashing for Collaborative Web Caching," Proc. World Wide Web Conf., pp. 633-644, 2000.

[7] Glen Jeh , Jennifer Widom, "Scaling personalized web search," Proceedings of the 12th international conference on World Wide Web, 2003.

[8] Kathleen R. McKeown, Noemie Elhadad, Vasileios Hatzivassiloglou, "Leveraging a common representation for personalized search and summarization in a medical digital library," Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries, 2003.

장 수 민

1997년 목포대학교 전산통계학과(학사) 1999년 충북대학교 정보통신공학과(석사). 2007년 충북대학교 정보통신공학과(박사). 2007년 충북대학교 초빙전임강사 2007년~현재 충북대학교 BK21 Post. Doc 관심분야는 게임서버, 정보검색, 분산 객체 컴퓨터 등

서 광 석

2008년 충북대학교 정보통신공학과(학사). 2008년~현재 충북대학교 정보통신공학과 석사과정. 관심분야는 데이타베이스시스템, 정보검색, 분산 객체 컴퓨터 등

유 재 수

1989년 전북대학교 컴퓨터공학과(학사)
1991년 한국과학기술원 전산학과(석사)
1995년 한국과학기술원 전산학과(박사)
1996년 충북대학교 전기전자공학부 부교수. 2006년~현재 충북대학교 전기전자공학부 교수. 관심분야는 데이타베이스시스템, 멀티미디어 데이타베이스시스템, 정보검색, 분산 객체 컴퓨터 등