Regular paper http://dx.doi.org/10.13067/JKIECS.2015.10.2.275

NAND 플래쉬메모리에서 고정그리드화일 색인의 성능 평가

김동현*

Performance Evaluation of Fixed-Grid File Index on NAND Flash Memory

Dong-Hyun Kim*

요 약

NAND 플래쉬메모리는 전원이 꺼져도 데이터를 유지할 수 있고 저비용, 고용량의 특징을 가지기 때문에 휴대용 기기에서 많이 활용되고 있다. 플래쉬메모리에서 대용량의 데이터를 효과적으로 처리하기 위하여 색인을 필수적으로 사용해야 한다. 그러나 플래쉬메모리는 쓰기 연산의 비용이 크고 덮어쓰기 연산을 지원하지 않기 때문에 기존의 디스크기반 색인을 사용하면 성능이 저하되는 문제가 발생할 수 있다. 본 논문에서는 플래쉬메모리에서 고정그리드화일 색인을 구현하고 다양한 조건에서 성능 평가를 수행한다. 이를 위하여 질의 연산과 변경 연산의 비율에 따른 평균 수행 시간을 측정한다. 그리고 디스크에서의 수행 시간과 비교한다.

ABSTRACT

Since a NAND-flash memory is able to keep data during electricity-off and has small cost to store data per bytes, it is widely used on hand-held devices. It is necessary to use an index in order to process mass data effectively on the flash memory. However, since the flash memory requires high cost for a write operation and does not support an overwrite operation, it is possible to reduce the performance of the index when the disk based index is exploited. In this paper, we implement the fixed grid file index and evaluate the performance of the index on various conditions. To do this, we measure the average processing time by the ratio of query operations and update operations. We also the compare the processing times of the flash memory with those of the magnetic disk.

키워드

색인, 플래쉬메모리, 고정그리드화일, 버퍼사용색인기법, 미버퍼사용색인기법

Index, Flash Memory, Fixed Grid File, Buffer Based Index, Non-buffer Based Index

Ⅰ. 서 론

플래쉬메모리는 비휘발성 메모리 칩을 이용하는 저장장치의 일종으로 전원이 꺼지더라도 저장 내용을 그대로 유지하고 있으며 휴대성이 좋은 특성을 가지고 있다. 또한 저비용, 작은 크기인 반면에 고용량의 데이터를 저장할 수 있는 특징이 있다. 따라서 USB 메모리와 같은 휴대용 저장장치로서 주로 사용되었으나 최근 스마트기기의 발전에 따라 스마트기기에 부착되어 활용되고 있는 실정이다[1], [2].

* 동서대학교 컴퓨터정보공학부(pusrover@dongseo.ac.kr)
접수일자 : 2014. 11. 20 심사(수정)일자 : 2015. 01. 16 게재확정일자 : 2015. 02. 09

플래쉬메모리는 논리회로의 종류에 따라 NOR형과 NAND형으로 분류되어 진다. 범용적으로 사용되는 NAND형 플래쉬메모리는 페이지와 블록단위로 데이터 접근을 허용하며 하나의 페이지는 512바이트로 구성된다. 데이터에 접근하기 위하여 기존의 디스크기반 저장장치와는 달리 전기신호를 사용하기 때문에 모든 저장영역에서 균일한 접근 속도를 보장하지만 쓰기 연산의 속도가 읽기 연산의 속도보다 매우 느린 특징을 가진다. 특히 동일 저장 주소에 덮어쓰기 연산을 지원하지 않고 기존의 페이지는 무효화시킨 다음에 자유 영역의 다른 페이지에 저장한다. 따라서 덮어쓰기 연산의 속도가 매우 느린 단점이 있다[3-5].

과거의 스마트기기에서는 연산장치의 처리 능력이 부족하기 때문에 처리하는 데이터의 양도 적었다. 따라서 소용량의 플래쉬메모리로도 충분하였다. 그러나 최근 스마트기기의 처리 능력이 매우 향상됨에 따라 처리하는 데이터의 양도 매우 증가하였다. 따라서 대용량 데이터를 신속히 처리하기 위하여 플래쉬메모리에서 구축된 색인이 필요하다. 그러나 기존의 디스크기반 색인구조를 플래쉬메모리에 적용하면 플래쉬메모리의 특성 때문에 색인의 성능이 저하되는 문제가 발생할 수 있다.

본 논문에서는 플래쉬메모리상에서 디스크기반 색인 성능을 파악하기 위하여 다차원 색인의 성능을 평가한다. 특히 다차원 색인구조 중에서 범용적으로 사용되고 있는 고정그리드파일 색인구조를 구현하고 성능 평가를 실시한다. 평가를 위하여 다양한 스케일, 버켓 그리고 데이터 집합에 대하여 평균 수행 시간을 측정하고 이를 자기디스크에서의 평균 수행 시간과 비교한다.

본 논문의 구성은 다음과 같다. 2장에서는 플래쉬메모리와 관련된 연구 동향을 기술하고 3장에서는 실험 조건과 실험 결과에 대하여 기술한다. 마지막으로 4장에서는 결론을 기술한다.

II. 관련 연구

2.1 NAND 플래쉬메모리

NAND 플래쉬메모리는 여러 개의 페이지와 블록으로 구성된다. 하나의 페이지는 512바이트와 16바이트의 여유영역으로 구성되며 하나의 블록은 32개의 페이지로 구성된다. NAND 플래쉬메모리는 구성 논리회로에 따라 다음과 같은 특성을 가지게 된다. 첫 번째는 기계적인 대기 시간이 없다. 플래쉬메모리는 전기 신호를 사용하여 셀의 데이터를 접근하기 때문에 모든 페이지의 데이터 접근을 위한 속도가 균일하다. 두 번째는 읽기와 쓰기의 속도가 다르다. 통상적으로 읽기 연산의 속도가 쓰기 연산의 속도보다 두 배이상 빠르다.

세 번째는 동일 위치에 덮어쓰기(overwrite)를 지원하지 않는다. 동일 위치에 데이터를 덮어쓰기를 할 경우에 새로운 데이터는 자유 영역에 위치한 다른 페이지에 저장하지만 자유 영역의 페이지가 부족하면 무효화 페이지를 초기화시키는 지움(erase)연산을 수행한다. 지움 연산은 페이지 단위가 아닌 블록 단위로 수행한다. 마지막으로 각 블록의 지움 연산 횟수는 제한이 있다. 통상적으로 1,000,000번의 지움 연산을 수행할 수 있으며 그 이후에는 해당 블록을 사용할 수 없다.

2.2 버퍼 사용 색인 기법

Chin H. W. 등.[1]은 B-tree를 이용한 BFTL(B-tree Flash Translation Layer) 기법을 제안하였다. BFTL은 기존 B트리의 구조 및 분할/합병 알고리즘을 사용하면서 노드의 엔트리 정보들을 저장하기 위하여 색인유닛을 사용한다. 새로운 키값의 삽입 또는 삭제 연산이 발생하면 색인유닛을 구성하고 플래쉬메모리에 바로 쓰지 않고 유보버퍼(reservation buffer)에 임시로 저장한다. 유보버퍼에 여유 공간이 없으면 완료정책(commit policy)에 따라 비로소 플래쉬메모리에 쓰기 연산을 수행한다.

Jung H.N 등.[2]은 BFTL 기법을 개선한 BOF(B tree On Flash memory) 기법을 제안하였다. BOF는 동일노드에 속하는 유보버퍼의 색인유닛들은 플래쉬메모리의 동일 페이지에 저장한다. 따라서 한 페이지에 하나의 색인 노드 정보만을 저장하기 때문에 한 번의 읽기 연산을 통하여 한 노드의 정보를 읽을 수 있고 노드변환 테이블을 사용하지 않는다.

Hyun S. L. 등.[6]은 트리 색인의 일종인 MR-tree를 적용하기 위한 기법을 제시하였다. Siwon Byun 등[7]에서는 플래쉬메모리의 쓰기 연산 횟수를 줄이기

위하여 색인의 수정, 즉 노드의 삽입/삭제 연산을 모아서 일괄 처리하는 기법을 사용한다. 노드의 삽입/삭제 연산의 내용을 메인메모리 버퍼에 모은 다음에 플래쉬메모리의 페이지 크기가 되었을 때 비로소 플래쉬메모리에 저장한다.

2.3 버퍼 비사용 색인 기법

Siwoo B. 등.[7]은 단방향재쓰기(one-way rewrite) 기법을 활용한 F-tree(Flash memory based Tree)를 제안하였다. 단방향재쓰기 기법은 쓰기 연산의 특성을 이용하여 쓰기 연산이 수행된 페이지만 지움 연산을 수행하지 않고 특정 비트의 1의 값을 0으로 재쓰기 하는 기법이다. 재쓰기 기법을 활용하여 F-tree는 각 엔트리의 정보를 압축한다. 만약 노드에 첫 쓰기 연산이 발생하면 노드 자료를 압축하여 빈 공간을 확보한다. 그리고 동일노드에 대하여 두 번째 쓰기 연산 즉, 덮어쓰기 연산이 발생하면 비어있는 공간에 재쓰기 연산을 수행하여 지움 연산의 횟수를 줄인다.

Yinan L. 등.[8]은 병합 기법을 사용하는 FD-tree를 제안하였다. FD-tree는 여러 레벨인 L_0부터 L_i로 나누어지는 런으로 구성된다. 만약 삽입연산이 발생하면 L_0에 먼저 삽입하며 상위레벨인 L_i의 런이 가득차면 L_i 레벨 런의 모든 엔트리를 하위레벨 L_{i+1}의 런으로 이동하면서 병합연산을 수행한다. 병합연산 후에 새로 생성된 L_{i+1} 레벨의 런은 L_i레벨 런의 기존 엔트리를 모두 포함하고 있으며 L_i 레벨 런의 외부경계도 새로 생성한다.

Mohamed S. 등.[9]은 새로운 색인구조를 제시하지 않고 기존의 B-tree, R-tree를 플래쉬메모리에서 사용하기 위한 프레임워크인 FAST(Flash Aware Spatial Tree index structure)를 제안하였다. 메인메모리 버퍼는 시스템 오류가 발생할 경우 색인의 연속성을 보장할 수 없기 때문에 FAST는 트리변경테이블과 로그파일을 사용하였다.

III. 성능실험

3.1 실험 환경

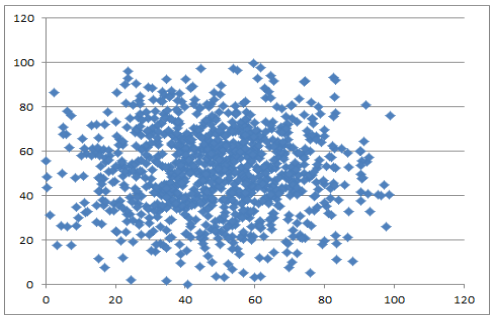


그림 1. 가우시안분포형 실험 데이터
Fig. 1 Gaussian distribution data for evaluation

실험을 위한 공간 데이터는 고정그리드 색인에 적합한 점 데이터집합을 사용한다. 데이터 집합의 크기는 3가지 종류로 구분하였으며 데이터 분포는 그림 1과 그림 2와 같이 가우시안분포형과 균등분포형인 2가지 종류를 사용하였다.

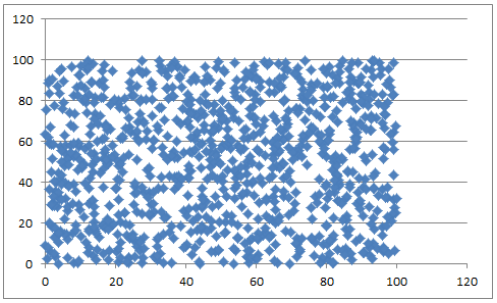


그림 2. 균등분포형 실험 데이터
Fig. 2 Uniform distribution data for evaluation

표 1. 실험 조건
Table 1. Evaluation condition

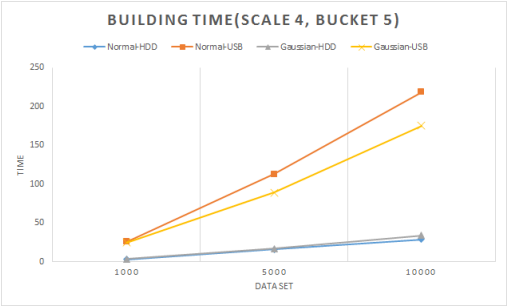
Title	Contents
Storage	Magnetic Disk, USB
Data Distribution	Gaussian, Uniform
Data Set Size	1,000, 5,000, 10,000
Scales	4×4, 32×32, 64×64, 256×256, 12×512
Bucket Size	1, 2, 5, 10, 21

고정그리드 색인의 성능은 공간을 분할하는 스케일의 수와 좌표 값을 저장하는 데이터 버켓의 크기에 따라 결정된다. 따라서 본 논문에서는 표 1과 같이 5

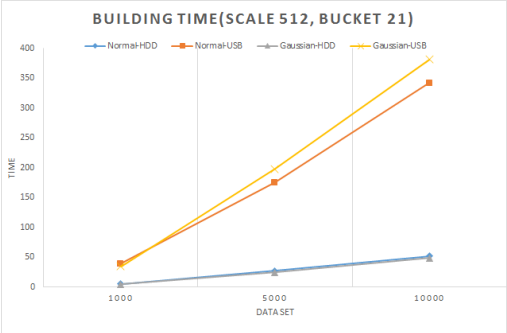
가지 종류의 스케일과 버켓의 크기를 이용하여 실험을 진행한다.

실험을 위한 저장장치는 자기디스크와 USB 메모리인 2가지 종류를 사용하였다. 각 장치의 접근 속도는 4k크기의 데이터에 대하여 자기디스크는 0.443(r)/1.085(w)(Mbyte/sec)이고 USB 메모리는 4.880(r)/0.191(w)(Mbyte/sec)이다.

3.2 실험 결과



(a)

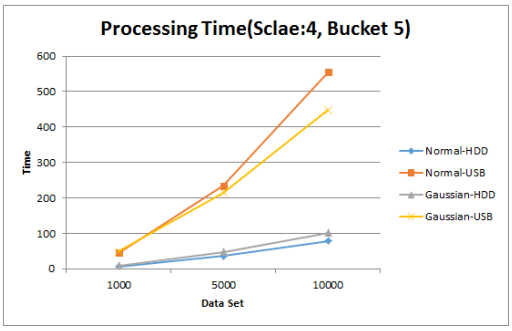


(b)

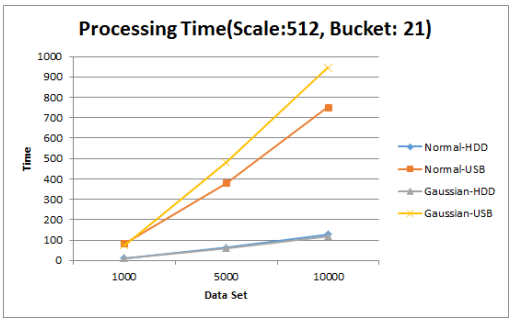
그림 3. 색인구축시간
Fig. 3 Index building time

그림 3은 자기디스크와 USB 플래쉬메모리에서 색인을 구축하기 위해 소요되는 시간을 비교한 그래프이다. 그래프의 가로축은 데이터집합의 종류를 나타내고 세로축은 각 데이터 집합에 대하여 소요된 시간이다. 그림 3의 (a)는 스케일이 4×4이고 버켓의 크기가 5인 경우이고 (b)는 스케일이 512×512이고 버켓의 크기가 21인 경우이다. 그림 3의 (a)에서 보듯이 균등분포형 데이터 환경에서 플래쉬메모리는 자기디스크에 비하여 11%~15%의 성능을 보여준다. 가우시안분포형은 13%~19%의 성능을 보여준다. 그림 3의 (b)에서 보듯이 균등데이터에서 플래쉬메모리는 13%~15%의 성능을 보여주고 가우시안데이터에 대해서는 13%~14%의 성능을 보여준다.

그림 4는 질의 연산과 변경 연산의 비율이 1:9로 변경 연산의 비율이 매우 높은 환경에서 연산 시간을 비교한 그래프이다. (a)는 스케일이 4×4이고 버켓의 크기가 5인 경우이고 (b)는 스케일이 512×512이고 버켓의 크기가 21인 경우를 보여준다. 그림 4(a)의 환경에서 균등 데이터인 경우에 플래쉬메모리는 14%~16%의 성능만을 보여주고 가우시안 데이터인 경우에는 18%~23%의 성능을 보여준다. 그림 4(b)의 환경에서 균등 데이터는 15%~17%의 성능을 보여주고 가우시안 데이터는 13%~15%의 성능을 보여준다. 이는 버켓의 크기가 21이 될 때 플래쉬메모리의 페이지 크기와 동일해지기 때문에 쓰기 연산의 성능이 개선되고 있지만 가우시안 데이터는 데이터가 집중되기 때문에 버켓의 크기가 커지더라도 성능이 저하되는 것으로 판단된다.



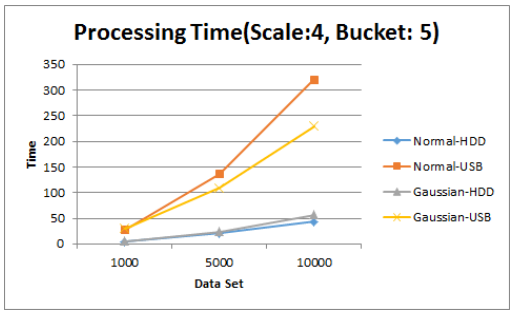
(a)



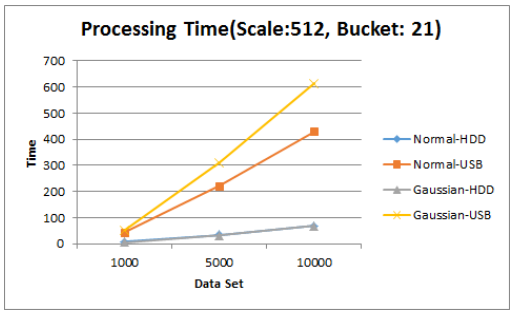
(b)

그림 4. 질의 : 변경 = 1 : 9에서 수행시간
Fig. 4. Processing time at query/modification = 1 : 9

그림 5는 질의 연산과 변경 연산의 비율이 5:5로 균등하게 수행되는 환경에서 연산 시간을 비교한 그래프이다. (a)와 (b)는 각각 스케일 4×4, 버켓 5와 스케일 512×512, 버켓 21인 환경을 보여준다. 그림 5(a)의 환경에서 플래쉬메모리는 균등 데이터에 대하여 13%~14%의 성능을 보여주고 가우시안 데이터에 대하여 18%~25%의 성능을 보여준다. 그리고 그림 5(b)의 환경에서 균등 데이터에 대하여 16%~19%의 성능을 보여주고 가우시안 데이터에 대하여 10%~12%의 데이터를 보여준다.



(a)

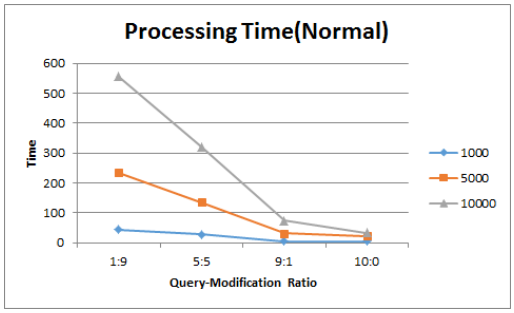


(b)

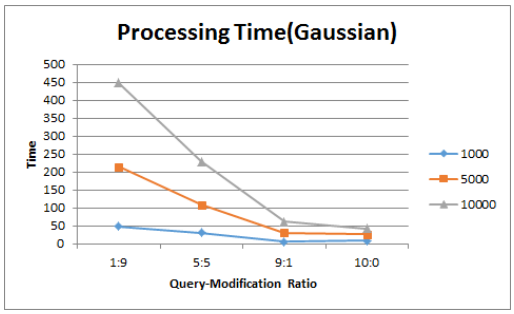
그림 5. 질의 : 변경 = 5 : 5에서 수행시간
Fig 5. Processing time at query/modification = 5 : 5

그림 6은 질의 연산의 비율이 변경연산에 비하여 배우 높은 9:1 조건에서 수행 시간을 비교한 그래프를 보여준다. 그림 6(a)에서 보듯이 스케일 4×4, 버켓 5인 환경에서 균등데이터는 15%~18%의 성능을 보여주고 가우시안 데이터는 19%~22%의 성능을 보여준다. 그림 6(b)에서 보듯이 스케일 512×512, 버켓 21인 환경에서 균등 데이터는 15%~17%의 성능을 보여주고 가우시안 데이터는 13%~15%의 성능을 보여준다. 이는 질의 연산의 비율이 증가하더라도 플래쉬메모리에서 변경 연산의 비용이 배우 크기 때문에 성능 개선의 효과가 미비하기 때문이다.

그림 7은 질의 연산의 비율에 따라서 플래쉬메모리에서의 수행시간을 비교한 그래프이다. 그림 7의 (a)와 (b)에서 보듯이 변경 연산의 비율이 줄어들 때 수행시간이 매우 개선되어 지고 있는 것을 볼 수 있다. 스케일 4×4, 버켓 1인 환경에서 균등 데이터는 평균 23%~58%로 개선되고 있고 가우시안 데이터는 27%~68%로 매우 개선된다. 이는 변경 연산의 비용이 질의 연산에 비하여 매우 크기 때문이다.



(a)



(b)

그림 6. 질의 비율에 따른 수행시간
Fig 6. Processing time by query ratio

실험 결과 그래프에서 살펴보듯이 플래쉬메모리에서 색인을 구축한 경우에 쓰기 연산의 비용이 매우 크기 때문에 질의 연산의 비율이 높아지더라도 자기 디스크에 비하여 성능이 매우 떨어지는 것을 볼 수 있다. 특히 색인은 동일한 객체가 변경될 경우에 덮어쓰기를 수행해야 한다. 그러나 플래쉬메모리는 덮어쓰기를 지원하지 않고 지움 연산을 한 후에 쓰기 작업

을 수행해야 하기 때문에 연산의 부하가 더욱 커지는 것을 확인할 수 있다. 그러나 버켓의 크기가 플래쉬메모리의 페이지 크기에 근접한 경우에는 연산 속도의 개선이 이루어지고 있기 때문에 고정그리드 색인의 성능 개선을 위해서는 버켓의 크기를 페이지의 크기와 동일하게 유지할 필요가 있다.

IV. 결 론

플래쉬메모리는 전원이 꺼지더라도 데이터를 유지할 수 있고 휴대성이 좋기 때문에 USB 메모리와 같은 휴대용 저장장치로서 많이 활용되고 있어, 대용량 데이터를 플래쉬메모미레 효과적으로 저장하고 처리하기 위한 색인에 대한 연구가 필요하다. 그러나 기존의 디스크기반 색인을 플래쉬메모리에 적용하면 플래쉬메모리의 특성 때문에 색인의 성능이 저하되는 문제가 발생할 수 있다.

본 논문에서는 플래쉬메모리에서 공간분할 색인의 일종인 고정그리드화일 색인을 구현하고 성능을 평가한다. 이를 위하여 데이터 집합과 스케일 그리고 버켓의 크기 등 다양한 조건을 적용하여 성능 평가를 수행하였다. 실험 결과를 보면 플래쉬메모리는 쓰기의 비용이 크고 덮어쓰기가 지원되지 않기 때문에 질의 연산의 비율이 높더라도 색인 변경 연산이 발생하면 디스크기반에 비하여 성능이 매우 저하되고 있는 것을 볼 수 있다. 향후 연구로는 고정그리드화일 색인에서 쓰기의 연산 횟수를 감소하고 덮어쓰기 연산을 수행하지 않기 위한 기법에 대한 연구가 필요하다.

감사의 글

본 연구는 2013년도 동서대학교 학술연구조성비 지원으로 수행되었음.

References

[1] Chil-Hsien Wu, Li-Pin Chang, Tei-Wei Kuo, “An Efficient B-Tree Layer for Flash Memory Storage System,” *Real-Time and Embedded Computing Systems and Applications,* LNCS 2968, Springer, 2003, pp. 409-430.

[2] Junghyun Nam, Dong-Joo Park, “The Efficient Design and Implementation of the B-Tree on Flash Memory,” *Proc. of the 25th Korea Institute of Information Scientists and Engineers Fall*, 2005, pp. 55-57.

[3] Yong-Yoong Chai, “An Analog Content Addressable Memory implemented with a Winner-Tak-All Strategy,” *The Journal of the Korea Institute of Electronic Communication Sciences*, vol. 8, no. 1, 2013, pp. 105-111.

[4] Yong-Yoong Chai, “An Analog Memory Fabricated with Single-poly Nwell Process Technology,” *The Journal of the Korea Institute of Electronic Communication Sciences*, vol. 7, no. 5, 2012, pp. 1061-1066.

[5] Chi-Yeon Kim, “A Range Query Method using index in Large-scale Database Systems,” *The Journal of the Korea Institute of Electronic Communication Sciences*, vol. 7, no. 5, 2012, pp. 1095-1101.

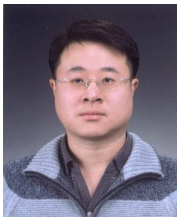
[6] Hyun Seung Lee, Ha Yoon Song and Kyung-Chang Kim, "Performance of Index trees on Flash Memory,” *Intl. MultiConf. on Computer Science and Information Technology*, 2007, pp. 725-734.

[7] Siwoo Byun, Moonhaeng Huh, Hoyoung Hwang, “An Index rewriting schemes using compression for flash memory database system,” *Journal of Information Science*, 2007, pp1-18.

[8] Yinan Li, Bingsheng He, Qiong Luo, Ke Yi, “Tree Indexing on Flash Disk,” *Proc. of ICDE*, 2009, pp. 1303-1306.

[9] Mohamed Sarwat, Mohamed F. Mokbel, Xun Zhou, Suman Nath, “FAST : A Generic Framework for Flash-Aware Spatial Trees,” *Intl. Conf. on Advances in Spatial and Temporal Databases*, 2011, pp. 149-167.

저자 소개



김동현(Dong-Hyun Kim)

1995년 부산대학교 컴퓨터공학과 졸업(공학사)
1997년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)
2003년 부산대학교 대학원 컴퓨터공학과 졸업(공학박사)
2004년~현재 동서대학교 컴퓨터정보공학부 교수
※ 관심분야 : 데이터베이스, 공간 데이터베이스, GIS, 센서데이터베이스