2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

MiDAS-Ⅲ에서 CIR-Tree를 위한 효율적인 벌크로딩 알고리즘의 설계

피준일, 송석일, 유재수
충북대학교 정보통신공학과
E-mail : {mouse76, prince, yjs}@pretty.chungbuk.ac.kr

Design of an Efficient Bulk Loading Algorithm for CIR-Tree on MiDAS-Ⅲ

Jun Il Pee, Seok Il Song, Jae Soo Yoo
Dept. of Computer and Communication, Chungbuk National University

요약

이 논문에서는 고차원 색인 구조인 CIR-트리를 위한 효율적인 벌크로딩 알고리즘을 설계하고 구현한다. 벌크로딩 기법은 대량의 고차원 데이터가 색인 구성 시 함께 주어지는 경우 색인의 구성을 빠르게 하고 구축한 색인의 검색 성능을 향상시킨다. CIR-트리는 변별력 있는 일부 차원만 이용해서 비단말노드의 엔트리를 구성하기 때문에 엔트리 크기가 일정하지 않다는 특징이 있으며 이는 비단말 노드의 팬아웃을 높이고 탐색 성능을 향상시키는 효과가 있다. 기존에 다차원 및 고차원 색인구조를 위한 벌크로딩 기법이 제안되었지만 이러한 CIR-트리의 특징을 제대로 살릴 수 있는 방법은 없다. 따라서 이 논문에서는 기존의 벌크로딩 알고리즘을 개선하면서 CIR-트리의 특징을 효과적으로 색인 구성에 반영할 수 있는 알고리즘을 제안한다. 또한 이를 BADA-Ⅲ의 하부 저장 시스템인 MiDAS-Ⅲ에서 구현하고 다양한 실험을 통해 그 성능을 입증한다.

1. 서론

최근 원격진료 시스템, 지리정보 시스템, 이미지 데이터베이스 시스템, 멀티 미디어 데이터베이스 시스템 등에서 내용기반을 보다 효과적으로 수행하기 위한 연구가 활발하게 진행되어 왔다. 이런 응용분야의 특성상 색인 구성 시 대량의 정적인 데이터가 주어지는 것이 보통이며 이런 경우 대량의 정적인 데이터를 하나씩 삽입하는 것은 색인 구성 시간이 매우 느리고 저장공간 활용률을 저하시킨다. 이러한 특징은 검색성능의 저하로 이어진다. 이런 문제를 해결하기 위해 주어진 데이터 집합을 미리 알고 있다는 점을 이용하여 구성 시간이 보다 빠르고 탐색 성능을 향상시킬 수 있도록 색인을 구축하는 벌크로딩 알고리즘이 제안되었다.

다차원, 고차원 색인구조를 위한 기존의 벌크로딩 알고리즘은 NX, HS, STR과 같은 정렬을 하는 방법과 UBBT나 버퍼-트리를 이용하는 방법과 같은 정렬을 하지 않는 방법으로 나눌 수 있다. 하지만 UBBT를 제외한 나머지 벌크로딩 알고리즘은 삽입시간측면이나 검색성능 측면을 모두 만족시키지는 못한다. 또한 고차원 색인 구조인 CIR-트리[1]는 팬-아웃을 증가시키기 위해 비단말 노드의 엔트리 구성 시 활성차원과 비 활성차원을 사용하게 되며 엔트리들의 크기가 모두 다를 수 있다. 이러한 특징으로 인해 UBBT를 포함한 기존의 벌크로딩 알고리즘을 CIR-트리에 그대로 적용하는 것은 문제가 있다.

※ 본 연구는 과학재단 특정기초과제(과제번호 : 1999-1-303-007-3) 연구비지원에 의하여 수행되었음.

이 논문에서는 기존의 벌크로딩 알고리즘을 개선하고 CIR-트리의 특징을 효과적으로 수용할 수 있는 벌크로딩 알고리즘을 설계한다. 또한 이를 BADA-Ⅲ의 하부 저장 시스템인 MiDAS-Ⅲ상에서 구현하고 다양한 실험을 통해 우수성을 입증한다. 이 논문의 구성은 다음과 같다. 2장에서 CIR-트리와 기존의 벌크로딩 알고리즘에 대해서 살펴보고 3장에서 제안하는 벌크로딩 알고리즘에 대해서 설명한다. 4장에서 CIR-트리를 위한 벌크로딩 알고리즘을 구현하여 성능 평가한 결과에 대해 기술하고 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1. CIR-트리

내용 기반 이미지 검색을 위해 제안된 고차원 색인 구조들은 차원이 증가할수록 색인 구조의 검색 성능이 현저하게 떨어지는 차원의 저주 현상이라는 문제점을 가지고 있다. 이런 문제점을 해결하기 위해 많은 색인 구조들이 제안되었으며 그중 하나가 CIR-트리이다. CIR-트리의 특징은 다음과 같이 정리할 수 있다. 첫째, 비단말 노드들에는 분별력이 있는 차원만으로 MBR을 구성하여 팬-아웃을 증가시킨다. 둘째, 겹침 영역을 최소화하기 위한 노드 분할 방법을 제공하며 수퍼 노드 개념을 도입한다. 셋째, 이미지 특징들의 군집화를 위해 무게 중심을 이용하는 보다 개선된 재삽입을 수행한다.

벌크로딩에서는 트리의 형태를 미리 계산해야 하는데 이때

색인 노드에 들어가는 엔트리의 수를 파악하는 것이 필요하다. 그러나 CIR-트리에서는 비활성 차원의 수가 엔트리마다 다르므로 정확한 개수 및 트리의 형태를 파악할 수 없다. 따라서 벌크로딩 알고리즘 설계 시 이를 고려해야 한다.

2.2. 기존의 벌크로딩 알고리즘

기존의 벌크로딩 알고리즘은 크게 정렬을 하는 방법과 하지 않는 방법으로 나눌 수 있다.

먼저 정렬을 하는 방법으로는 NX[2], HS[3], STR[4], TGS[5] 등이 있다. 정렬을 수행하는 벌크로딩 알고리즘들은 색인의 구축 후 탐색 성능이 향상된다는 장점이 있지만 대량의 데이터에 대해 정렬을 수행해야 하므로 색인의 구성 시간이 오래 걸린다는 단점이 있다.

정렬을 수행하지 않는 벌크로딩 알고리즘으로는 버퍼 트리를 이용하는 방법[6]과 UBBT(Unbalanced Bisection Bulk-loadging Technique)[7]가 있다. [6]의 벌크로딩 알고리즘은 삽입 시간은 정렬방법보다 빠르지만 데이터 집합의 특성을 효과적으로 반영할 수 없어서 탐색 성능을 향상시킬 수 없다는 단점이 있다. 이에 반해 UBBT는 데이터들을 정렬하지 않고 이분하는 방법을 제안하였으며 기존의 벌크로딩 알고리즘들중 삽입 시간측면이나 검색 성능 측면에서 가장 우수한 성능을 나타내고 있다.

하지만 UBBT에는 구현 시 몇 가지 고려해야할 문제점이 있다. 먼저 분할 범위 계산 시 잘못된 분할 범위를 갖는 경우가 발생하는데 이는 트리의 형태 계산 시와 실제 이분 시 다른 트리 형태를 적용하기 때문이다. 또 다른 고려사항은 실제 이분 시에 발생한다. UBBT에서 이분은 퀵 정렬 알고리즘을 변형시켜서 사용하고 있다. 데이터 집합 전체를 정렬하는 것이 아니고 단지 피벗값을 기준으로 특정 벡터의 분할 차원 값이 작은 것과 크거나 같은 것으로 이분한다. 하지만 불균등한 분포를 갖는 데이터 집합에 대해서는 피벗값과 같은 값을 갖는 데이터들이 분할 범위에 걸쳐 있을 수 있다. 이 경우 피벗값은 결코 분할 범위에 포함될 수 없으며 이분은 멈출 수 없다. 또한 앞서 언급했듯이 CIR-트리 중 변별력 있는 일부차원만을 사용한다는 특징으로 인해 정확한 트리의 형태를 미리 예측할 수 없으므로 UBBT를 수정없이 그대로 적용하는 것은 문제가 있다.

본 논문에서는 UBBT 알고리즘의 두가지 문제점을 해결하면서 CIR-트리의 특징을 효과적으로 지원하는 벌크로딩 기법을 설계하고 구현한다.

3. 제안하는 알고리즘

CIR-트리를 위한 벌크로딩 알고리즘은 주어진 데이터 집합을 이분을 통해 단말 노드를 생성하고 단말 노드가 생성되면 그 단말 노드의 MBR을 상위에 반영하여 상향식으로 색인을 생성하는 방식을 취한다.

CIR-트리를 위한 벌크로딩 알고리즘은 크게 3단계로 구성된다.

- 트리의 형태 계산 : 높이와 팬-아웃 수 계산
- 부 데이터 집합으로 분할 : 데이터 분할 전략 결정과 이분
- 색인 노드 생성

루트 노드가 완성되어 색인이 구성될 때까지 위의 세 단계를 반복 수행하게 되는데 각 단계에 대해서는 순서대로 살펴보도록 하겠다.

3.1. 트리 형태 (Tree Topology) 계산

벌크로딩을 통해 생성될 색인 구조의 형태를 미리 계산하는 단계이다. 계산을 통해 얻어지는 최종 형태로는 생성될 색인의 높이와 팬-아웃수 등이다. 팬-아웃수는 각 노드마다 들어갈 수 있는 엔트리 수를 말한다. 이때 단말노드와 색인 노드에 들어갈 수 있는 최대 엔트리 수를 계산해야 하며 노드 크기와 엔트리 크기로부터 구할 수 있으며 이를 각각 Cmax,data 와 Cmax,dir이라 한다. 또한 저장공간 활용률을 고려하여 평균적으로 들어갈 수 있는 엔트리 수를 Ceff,data와 Ceff,dir이라 한다. CIR-트리에서는 엔트리마다 비활성 차원의 수가 다르기 때문에 색인 노드에 들어갈 수 있는 엔트리의 크기와 개수 계산 시 비활성 차원과 활성 차원을 고려하여야 한다. 여기서 비활성 차원의 수는 현재 데이터 집합 전체가 갖는 비활성 차원의 수로 한다. 이로부터 트리의 형태 중 높이(h)와 팬-아웃(fan-out)을 (수식1)과 (수식2)를 통해 계산할 수 있으며 이정보는 데이터 집합 분할 시에 사용된다. 여기서 n은 데이터의 개수를 의미한다.

$$h = \lceil \log_{Ceff,dir}\left(\frac{n}{Ceff,data}\right) \rceil + 1 \quad \text{..... (수식 1)}$$

$$fanout(h,n) = \min\left(\lceil \frac{n}{C_{eff,data} \cdot C_{eff,dir}^{h-2}} \rceil , C_{\max,dir} \right) \quad \text{..... (수식 2)}$$

3.2. 부 데이터 집합으로 분할

부 데이터 집합으로 분할하는 과정은 데이터 분할 전략 수립 단계와 이분 과정을 반복 수행하며 원 데이터 집합을 앞서 구한 팬-아웃 수만큼의 부 데이터 집합으로 나누는 과정이다.

가) 데이터 분할 전략 수립

먼저 원 데이터 집합을 이분하기 위해서는 데이터 집합 분할 시 기준이 되는 분할 차원과 분할 위치를 결정해야 한다. 분할 차원은 최대 길이를 갖는 차원으로 선택한다. 이는 최대 길이를 갖는 차원을 분할 차원으로 선택하는 것이 질의 시 두 분할된 영역을 모두 찾아갈 확률이 더 적기 때문이다. CIR-트리의 특징을 고려하여 분할 차원 선택 시 활성 차원 중 최대 길이를 갖는 차원으로 선택한다. 분할 위치는 분할 비율에 따라 결정하게 되는데 비 균등 분할을 원칙으로 하며 정확한 분할 위치를 찾는 대신 보다 빠른 이분 수행을 위해 데이터 분할 범위를 둔다. 이럴 경우 앞서 제기했던 것처럼 UBBT의 잘못된 분할 범위의 계산 문제를 고려해야 한다. 여기서는 분할 범위 계산 시 부트리가 가져야 하는 최소한의 데이터 수를 고려함으로써 어느 한쪽으로 데이터들이 집중되는 것을 피하도록 하여 문제를 해결하고 있다. 부트리가 가져야 하는 최소한의 데이터 수는 단말 노드와 비단말 노드 모두 각 노드가 가져야 하는 최소한의 데이터 수, 즉 MinFill을 갖는 것으로 간주하여 계산한다.

나) 이분(Bisection)

원 데이터 집합을 두 개의 부 데이터 집합으로 이분하는 과정으로 데이터의 이분 과정은 높이 우선으로 수행되며 같은 레벨일 경우 왼쪽 부 데이터 집합에 대해 먼저 반복 이분을 수행한다. 전 단계에서 구한 트리의 형태와 데이터 분할 전략

을 바탕으로 주어진 데이터 집합을 선택된 차원으로 이분하며 외부 이분과 내부 이분으로 처리한다. 벌크로딩 시 빠른 이분 수행을 위해 일정량의 메모리를 사용하게 되는데 데이터들이 메모리로 모두 올라올 수 있는지 여부에 따라서 내부 이분 혹은 외부 이분으로 나누어 처리한다.

내부 이분의 경우 모든 데이터들을 메모리로 로드하여 이분을 수행하면 된다. 외부 이분의 경우에는 내부 이분 시 사용하는 메모리를 이용하여 주어진 분할 비율에 따라 피벗값을 결정하고 피벗값을 기준으로 데이터 집합을 이분한다.

제안하는 알고리즘에서 이분 시 또 다른 특징은 UBBT의 문제점을 해결하기위해 데이터 분할 범위 내에서 정상적인 이분이 가능하도록 피벗값과 같은 값을 갖는 데이터들을 별도로 관리한다는 것이다. 이분 후 원 데이터 집합은 피벗값보다 작은값, 같은값, 큰값을 가지는 세 개의 집합으로 분할되며 피벗값과 같은 값을 갖는 데이터 집합의 위치를 고려하여 데이터 분할 범위에서 이분되도록 보장한다. 또한 예기치 못한 비활성 차원의 발생으로 색인 노드 구성 시 발생할 수 있는 넘침을 해결하기 위해 (그림 1)처럼 차등 이분법을 적용한다. 하나의 색인 노드로 구성될 데이터 집합에 대해 처음 분할 시에는 MinFill을 만족하는 범위내에서 비균등 분할을 하며 그 이외에는 비균등 분할 비율을 적용한다. 1차 분할 시 사용했던 분할 전략은 색인 노드 구성 시 넘침이 발생하였을 경우 노드 분할에 사용된다. 따라서 비균등분할로 인한 검색성능 향상과 넘침 발생 시 겹침이 없는 노드 분할을 수행할 수 있다.

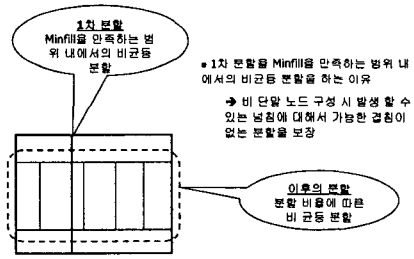

그림 1. 차등 이분법

3.3. 색인 노드(비 단말 노드) 생성

원 데이터 집합을 부 데이터 집합으로 반복해서 이분하면 단말노드를 생성하게 된다. 이렇게 단말노드가 생성되면 그 정보를 상위노드에 반영하여 색인을 구성하는 상향식 색인 구성 방식을 따른다. 비단말 노드 엔트리들을 상위 노드에 반영할 때 예기치못한 비활성 차원수의 증가로 인해 넘침 현상이 발생할 수 있다. 이는 트리의 형태 결정 시에 모든 엔트리의 비활성 차원을 고려한 것이 아니므로 발생한다. 만약 넘침이 발생하면 (그림 1)에서 1차 데이터 분할에 사용했던 분할기준을 적용하여 MinFill을 만족하면서 겹침이 없는 노드 분할을 수행한다. 하지만 어느 한쪽에 비활성 차원이 많이 생긴다면 그쪽은 여전히 넘침이 발생한다. 이럴 경우 수퍼노드로 확장해서 해결한다.

(그림 2)는 벌크로딩 알고리즘에 따라 수행되어 색인을 구성한 예이다. 그림의 상단 부분은 데이터 집합을 반복 이분하여 단말노드를 생성하는 모습이며, 하단 부분은 생성된 단말노드로부터 색인 노드를 생성하여 색인 구조를 완성하는 모습이다. 여기서 두 번째 레벨의 가장 오른쪽 노드를 구성할 때 넘침이 발생하였고, 분할을 통하여 해결하여 색인을 구성한 모습이다.

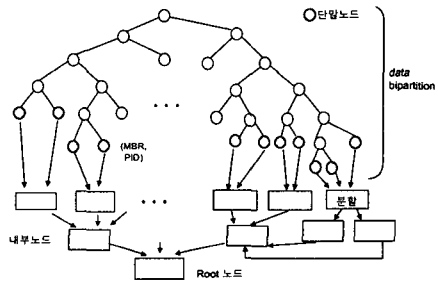

그림 2. 색인 구조 생성의 예

4. 성능평가

CIR-트리를 위한 벌크로딩 알고리즘은 바다-Ⅲ의 하부 저장 시스템인 MIDAS-Ⅲ에서 구현하여 다양한 성능 평가를 수행하였다. 구현에 사용된 시스템은 Solaris 2.7.x 운영 체제에 Sun Enterprise 250이며 메인 메모리의 크기는 1GByte로 벌크로딩 수행 시 내부 이분을 수행하기 위해 충분한 메모리를 획득할 수 있었다. 사용된 컴파일러는 gcc 2.8 이다

실험은 9차원의 실제 동영상 데이터 집합에 대해서 수행하였다. 먼저 벌크로딩 알고리즘으로 구축할 경우 저장공간의 절약 효과를 알아보기 위해 CIR-트리 구축에 사용된 페이지 수를 측정하였다. (그림 3)은 전체 9차원의 실제 동영상 데이터 집합에 대해 활성 차원 수를 5로 하고 데이터 수를 변화시켜 색인 구성 시간에 대한 성능 평가를 한 결과이다. 그림에서 보듯이 벌크로딩으로 색인을 구성할 경우 구성시간 측면에서 탁월한 성능을 나타내고 있다. (그림 3)에서 구성한 색인에 대한 K 최근접 질의를 수행한 결과가 (그림 4)에 나타나 있다. K 최근접 질의 수행 시 벌크로딩으로 색인을 구성하였을 경우 페이지 접근 수가 18~40% 정도 감소됨을 알 수 있다. 또한 (그림 5)에서와 같이 저장공간 활용률 측면에서도 벌크로딩 알고리즘이 개별 삽입에 비해 약 16%정도의 저장 공간 절약 효과가 있음을 알 수 있다. 이것은 저장공간 활용률을 80%로 보장하여 색인에 사용된 페이지들이 불필요하게 낭비되는 현상을 막았기 때문이다.

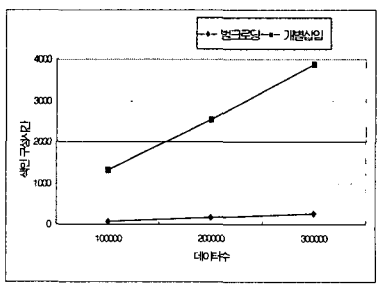

그림 3. 색인 구성 시간

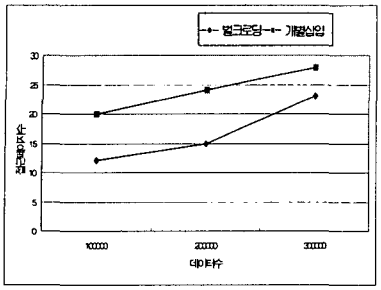

그림 4. K-최근접 질의 (K=10)

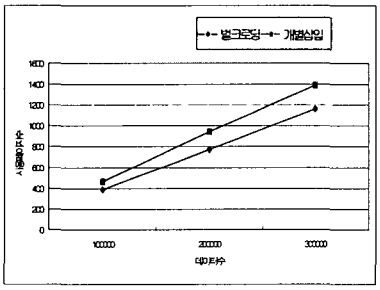

그림 5. 저장 공간 활용도

다음 실험에서는 X-트리에 적용한 UBBT와 성능 비교를 하였다. 여기서는 K가 10일때의 K 최근접 질의에 대한 성능비교만을 하였다. 제안하는 알고리즘이 MiDAS-Ⅲ라는 특수한 환경에서 구현되었고, UBBT는 UNIX의 파일 기반으로 X-트리에 구현되어 있기 때문에 색인 구성 시간 측면에서 직접적인 성능 비교를 할 수 없다. 사용된 데이터 집합은 10차원의 균등분포 데이터 50만개로 10만개에서 50만개까지 변화시키면서 실험을 하였다. 제안하는 방법과 UBBT로 생성된 색인 구조에 대해 K=10일 경우 최근접 질의에 대한 성능 비교 결과가 (그림 6)에 나타나 있다.

CIR-트리를 위해서는 전체 차원, 활성 차원 모두 10으로 하여 색인을 구성하였다. (그림 6)에서 보듯이 평균적으로 40%정도 제안하는 방법이 좋은 성능을 나타내고 있으며 데이터 수가 많은 경우에는 최고 4배 정도의 우수한 성능을 보이고 있다.

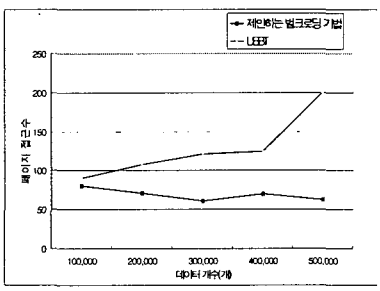

그림 6. K-최근접 질의

5. 결론

이 논문에서는 CIR-트리를 위한 벌크로딩 알고리즘을 제안하고 구현하였다. 그동안 고차원 색인 구조를 위한 벌크로딩 알고리즘들이 일부 제안되었지만 기존의 벌크로딩 알고리즘들이 색인 노드 구성 시 일부 차원만 사용하는 고차원 색인 구조에 적합하지 못하고 검색 성능 측면에서나 색인의 구성 시간 측면에서 좋은 성능을 보이지 못하였다. 이 논문에서는 UBBT 기법을 기반으로 하여 기존 알고리즘의 문제점을 해결하고 CIR-트리에 적합한 벌크로딩 기법을 설계하였다.

이 논문에서 CIR-트리를 위한 벌크로딩 기법을 설계할 때 고려했던 점들을 네 가지로 요약할 수 있다. 첫 번째, UBBT에서 제안한 트리 형태 계산을 통한 비균등 이분법을 기반으로 하며 보다 개선된 그리고 CIR-트리의 특징을 수용하는 벌크로딩 기법을 설계하였다. 두 번째, 분할 범위를 계산할 때 부 트리가 가져야 하는 최소 한계 데이터 개수를 고려하여 잘못된 분할 범위가 계산되는 문제점을 해결하였다. 세 번째, 이분을 수행할 때 분할 차원의 값이 피벗값과 같은 데이터들을 따로 모아서 관리함으로써 분할 범위내에서 이분이 가능하도록 보장한다. 네 번째, CIR-트리가 갖는 가변 MBR 특징을 수용하는 방법을 고려하였다.

제안한 벌크로딩 알고리즘은 바다 DBMS의 하부 구조인 MiDAS-Ⅲ에서 구현하고 성능 평가를 하였다. 성능 평가에서 보듯이 개별 삽입보다 색인의 구성 시간, 저장 공간 활용률, 그리고 검색 성능 측면에서 모두 우수함을 나타냈으며 UBBT 방법과의 탐색 성능 비교에서도 우수함을 타나냈다.

참고문헌

[1] 이석희, 송석일, 유재수, "내용기반 이미지 검색을 위한 고차원 색인구조," 한국정보과학회 데이터베이스 연구회 논문지, 제 14권, 제 4호, pp. 53-68, 1998

[2] Roussopoulos N. and Keifker D., "Direct Spatial Search On Pictorial Databases Using Packed R-Trees," In Proc. ACM SIGMOD, pp. 17-31, 1985

[3] Kamel I. and Falousos C., "On Packing R-Trees," In Proc. CKIM, pp. 490-499, 1993

[4] Leutenegger S. T., Lopez M. A. and Edgington J., "STR : A Simple and Efficient Algorithm for R-Tree Packing," In Proc. ICDE, pp. 497-506, 1997

[5] Garcia Y. J., Lopez M. A. and Leutenegger S. T., "A Greedy Algorithm for Bulk Loading R-Trees," In Proc. ACM GIS, pp. 47-57, 1998

[6] Van Den Bercken J. and Seeger B., Widmayer, "A General Approach to Bulk Loading Mutidimensional Index Structures," In Proc. VLDB, pp. 406-415, 1997

[7] Christian Böhm, "Efficient Bulk Loading of Large High- Dimensional Indexes," In Proc. Dawak , 1999