화일조직을 위한 인덱싱 기법의 성능 특성 비교

이 구 남*

요 약

이 논문은 자료접근 방법중 신속한 자료접근 및 갱신작업을 위해 일반적으로 많이 사용되는 인덱스(INDEX) 기법을 분석 비교함으로써 효율적인 데이타베이스 시스템을 위한 기초를 제공하고자 한다. 이를 위해 기존의 자료접근 기법들을 편의에 의해 자료 군집형태와 검출 형태에 따라 분류하고 인덱스 기법을 단일키와 다중키로 구분하였다. 단일키 인덱싱 기법을 동적 인덱스와 정적 인덱스 기법으로 나누어 비교 설명하고, 다중키 인덱스 기법중에서 K-d 트리, K-d-B 트리 및 역 화일과 그리드 화일의 자료 구조 특성을 비교 분석한 다음 각 기법들의 구조적 특성과 추출 질의어에 의한 일반적인 성능 특성을 비교 분석하였다.

Some Characteristics of the Performance in Comparison with Indexing techniques for File Organization

Gu Nam Lee*

Abstract

In this thesis, To provide the base of effective data access methods, performance of some indexing techniques used generally are compared. They are classified as primary key and multikey. For primary key method, made a comparative analysis on Static index,Dynamic index and Hashing. For multikey indexing method K-d tree, K-d-B tree, Inverted file and Grid file of which characteristics are compared. In many applications, multikey indexings are more requested, but are not supplied enough. So, to satisfy users' request - more fast, more exact and to be applied according to the trend of being huge database systems, it is requested more study about multikey data access methods .

Ⅰ. 서 론

정보의 생산량이 증가하고 또 축적되어 감에 따라 정보 이용분야의 범위도 넓어지고, 정보에 대한 요구도 다양해지고 있다. 이러한 다양하고 복잡한 정보 요구를 충족시키기 위해 대량의 데이타를 어떻게 체계적으로 조직하고 저장하여 효율적으로 처리 하느냐에 대한 연구가 일찍부터 논의되어 왔다.

특히 테이프와 디스크가 데이타를 저장하는 대중 매체로서 그 용량과 속도가 급속도로 대형화하고 신속화되면서 실세계에서 발생하는 데이타를 저장하고 관리하는 방식으로 데이타베이스 관리시스템(DBMS)을 이용하는 경향이 급속히 증가하고 있는

* 정회원 : 광주경안국민학교

추세이다. 다량의 자료를 유지,관리 하여야 하는 데이타베이스 시스템의 성능은 일반적으로 그 데이타베이스 시스템이 갖고 있는 자료 접근 기법에 영향을 받는다고 할 수 있다. 따라서 저장공간의 효율성과 성능이 우수한 자료 접근 구조의 연구가 중요하다고 할 수 있겠다.

이에따라 자료접근 방법중 신속한 자료접근및 갱신작업을 위해 일반적으로 많이 사용되는 인덱스(INDEX) 기법을 분석 비교함으로써 효율적인 데이타베이스 시스템을 위한 기초를 제공하고자 한다.

이를위해 기존의 자료접근 기법들을 편의에 의해 자료 군집형태와 검출 형태에 따라 분류하고, 인덱스 기법을 단일키와 다중키로 구분한다. 단일키 인덱싱 기법을 동적 인덱스와 정적 인덱스 기법으로 나누어비교 설명하고, 다중키 인덱스 기법중에서 K-d 트리, K-d-B 트리 및 역 화일과 그리드 화일의 자료 구조 특성을 비교 분석한 다음 각 기법들의 구조적 특성과 추출 질의어에 의한 일반적인 성능 특성을 비교 분석한다.

Ⅱ. 자료 접근 방법의 분류

일반적으로 자료접근 방법은 데이타가 저장되는 방법인 저장구조와 저장된 자료를 추출하는 탐색기법으로 구분된다. 저장구조는 저장된 자료로 접근할 수 있는 기본적인 접근 경로를 제공하며, 탐색기법은 저장 구조 내의 자료에 도달할 수 있도록 하는 접근 경로를 정의하는 알고리즘과 그에 필요한 작업들을 말한다.

■. 군집(CLUSTERING)방법에 의한 분류

자료접근 구조에 의한 데이타 화일, 즉 탐색공간은 여러개의 데이타 블럭으로 나누어지게 된다. 데이타 블럭들의 각 노드들은 특별한 군집(clustering) 특성을 갖는다.

일반적으로 세가지 종류의 특성으로 분류할 수 있다[2].

가) 연관 특성 : 데이타가 논리적으로 군집을 이루고 있는 특성.

나) 해쉬 특성 : 논리적인 키 값이 인공적인 함수의 적용에 의해서 다른 공간으로 변환되는 군집 특성이다.

다) 상대적 특성: 데이타 노드에서 임의구조를 갖는 군집특성이다.비 순차적 화일에서 이러한 특성을 사용한다.

이상의 자료 접근 구조에 따른 군집방법의 분류를, 편의상 단일키 접근 구조와 다중키 접근 구조로 나누어 분류하면 다음과 같다.

□. 단일키에 의한 자료 접근 구조

크게 나누어 연관(Associative) 특성과 해쉬(Hashed)방법으로 구분하며,해쉬 방법은 키값이 계산상의 변환을 거쳐 주소가 되고, 키의 자연적 순서는 상실하게 된다. 연관 방법은 키를 변환하지 않고 키들을 분류된 순서로 위치시킬 수있는 방법이다.

1) 연관(Associative)방법

이것은 비 인덱스-연관 방법과 인덱스-연관 방법으로 대별된다. 비인덱스-연관 방법은 구현이 간단하다는 장점이 있으나 동적 변화에 적당하지 않다. 레코드 탐색시간이 긴 반면 레코드 삽입 시간은 짧다. 이 방법에 속하는 화일로는 비정렬된 물리적 순차화일(Unsorted Phy-sycal Sequential file), 비정렬된 연결 리스트 화일(Unsorted Linke-d List file), 정렬된 물리적 화일(So2rted physical file),정렬된 연결 리스트 화일(Sorted Linked List file)등이 있다.

인덱스-연관(Associative Indexed) 방법은 키 부분에 의해 분류되는 화일로 동적인 확장, 신속한 갱신작업, 신속한 추출작업을 위해 인덱스를 사용한다. 트리 인덱스는 화일구조2의 균형(balance) 여부, 주기적인 재구성등에 의해 2차이가 생긴다. 이에 속하는 화일로는 B-트리,B(*2)-트리, B(+)-트리, Prifix B-트리, Digital B-트리2, DL-트리와 AVL-트리 등이있다.

2) 해쉬(Hashed) 방법

키를 변환하는 하는 가장 큰 이유는 균일치 않은 키 분포를 균일한 키 분포로 바꾸게 되며, 매우 큰 2키 저장 공간을 보다 적은 공간으로변환할 수 있기 때문이다. 화일의 크기가 변하는 화일에 사용할 수 있는 해쉬 방법은 해쉬 함수 자체를 변경 하는 방법과 변경하지 않는 방법 2가지로 구분된다. 전자에 해당하는 화일은 비트 인덱스(bit index)가 필요한 가상(Virtual) 해쉬 방법과 인덱스가 필요치 않은 선형 해쉬 방법이 있으며, 후자에 해당하는 화일에2는 동적해싱과 확장해싱 등이 있다.

□. 다중키에 의한 자료접근 구조

다중키 화일 구조는 다중키를 구성하는 모든 속성을 사용하여 탐색성능을 향상시키기 위한 구조이다.

1) 연관 방법

인덱스-연관 방법의 간단한 화일구조는 다중키를 단일키로 변환하여 단일키 화일 구조를 다중키 화일 2구조로 사용하는 방법 이다. 이 방법은 키 순서에 의한 단일키 화일 구조에서 효율적으로 사용될 수 있으나 다중키를 불균형하게 사용할 경우 탐색성능이 떨어지게 된다.

인덱스-연관 방법을 사용하는 데이타 구조에는 K-d 트리[4,13],K-d-B 트리[14] 등이 있다.

2) 해쉬 방법

다중키 해쉬 방법은 해쉬 함수가 변환된 키 공간이 균일한 분포를생성할 때, 완전 매치와 부분 매치 질의어 수행시 탐색성능이 최고가 된다. 이에 해당하는 화일 구조로 그리드 화일[12,15], 다중키 확장-해쉬, 역(Inverted) 화일 등이 있다.

■. 자료 검출 형태의 분류

접근 기법들을 사용자의 자료 검출 형태를 기준으로 분류하는 방법이다[16]. 검출 형태는 3 부류로 나2뉘게 되며, 각 부류에 속하는 방법은각 검출 형태에 최적으로 사용되는 방법이다. 이와 같이 분류할 경우 응답시간 요구 및 저장 공간의 제한을 최적으로 만족하는 접근 기법들을 고려하여 설계할 수 있다. 분류된 부류는 다음과 같다.

가) 전체 검출 또는 다수 검출

이 부류는 데이타베이스 내 레코드들의 10-100 %에 해당되도록 접근하는 형태이다. 순차적인 처리, 다량의 보고서 작성, 배치처리등이이에 해당된다.

나) 단일 검출

데이타베이스 내의 단일 레코드를 접근하는 검출 형태이다. 기본 키에 의해서 임의 접근 기법과 인덱싱기법을 이용하는 방식이 여기에 해당된다.

다) 일부 검출

전체 레코드의 0-10%에 해당하는 레코드를 검출하는 형태이다

□. 전체 검출 또는 다수 검출 구조

1) 물리적 순차(Physically Sequential)구조

가장 간단한 구조로, 저장장치의 연속적인 물리적 공간에 레코드를 순차적으로 저장하는 구조를 갖는다. 대용량의 순차처리 작업시효율적인 물리적 저장 구조이다. 레코드들이 저장장치 내에서 연속적으로 저장되어 있으므로, 인접한 데이타를 접근하는 순차 처리시 데이타 접근 시간을 최소화 할수 있다.

2) 연결된 순차(Linked Sequential)구조

물리적으로 연속되지 않은 데이타들에 대해서 논리적인 순차처리작업을 할 경우에 사용되며, 각 레코드들은 포인터로 연결되어 저장된다. 이진 탐색 방법은 적용될 수 없으며, 리스트로 연결된 각 레코드들은 다른 데이타 블럭에 위치할 수 2있다.

□. 단일 검출 구조

1) 직접 접근 구조

가능한 모든 1차 키 값에 별도의 레코드 저장 장소를 할당할 수있으면 간단한 키주소 변환 함수로

레코드를 한번의 블럭 접근에 의해 저장하거나 추출할 수 있다. 키 주소 변환 함수는 한 물리적 블럭에 저장된 레코드의 키 집합에 대해 상대적 블럭 번호를 할당하여 모든 데이타 추출 작업이 한번의 블럭 접근으로 완료되나 저장공간 사용은 모든 변환 2가능한 키 값들의 크기에 의해 결정된다. 그러므로 경우에 따라 저장 공간의 비용이 크게 된다.

2) 해쉬 구조 (ramdom access)

해쉬 방법은 직접 키값을 상대적 또는 물리적 주소(physical -address)로 변환하여 데이타에 접근하는 기법이다. 키값들과 주소를 N:1 로 변환하기 때문에 다수의 키들이 하나의 물리적 주소로 변환될 경우 충돌(collision)이 발생한다.

3) Full indexed(Indexed random)구조

화일 내에 존재하는 모든 레코드들에 대해서 인덱스를 만드는 방법이다. 일반적으로 신속한 탐색2을 위해 인덱스 간에 순서를 정할 수있으나 데이타 레코드 간에는 순서나 물리적 연속조건이 없다. 2각 데이타 레코드들이 서로 무관하게 저장되어 있으므로 갱신 비용이 적게드나 특정 순서에 의해 추출될 경우 성능이 떨어진다.

4) 인덱스 순차 구조

이 구조는 순차적인 접근과 인위적인 접근을 동시에 만족하기 위한 것이다. 순서가 정해진 인덱스와 순서가 정해진 데이타 레코드를 사용한다.데이타 공간은 일정한 크기의 블럭으로 나누어지며, 각 블럭마다 해당 블럭에 저장된 가장 큰 키 값과 해당 블럭 주소로 구성된 블럭 인덱스(block index)를 만든다. 화일 내의 순서는 블럭 인덱스를 이용하여 보존한다. 블럭 내에서는 데이타들이 순차적으로 저장되고 추출되는 기법이다.

□. 일부 검출 부류

1) 다중리스트 구조

이 구조는 한 데이타 종류의 같은 값을 가지는 데이타 레코드들을 리스트로 연결하는 연결된 순차구조(Linked Sequential Structure)의 집합을 기본 구조로 하고 있다. 연결 리스트들은 또한 다른 키에 신속하게 접근될 수 있도록 별도의 인덱스를 구성한다. 레코드 단계의 탐색은 순차적으로 수행이되며, 인덱스 단계의 탐색은 순차적으로 수행되거나 순서가 정해진 경우에는 이진탐색 또는 트리 구조 탐색에 의해 수행될 수 있다.

2) Cellular 다중 리스트

이 구조는 레코드와 레코드 단위로 연결되었을 뿐 아니라 각 레코드들이 저장되어 있는 블럭단위로 연결된 다중리스트(Multilist)구조이다. 잇점은 같은 종류의 레코드들로 블럭을 할당할 경우 주소 저장 공간을 줄일 수 있으며, 연결된 모든 레코드들을 접근할 경우 접근 시간을 줄이게 된다. 그러나 블럭에 일정 수준의 동형레코드를 저장하기위한 CPU 부대 비용이 소요되게 된다.

3) 역(Inverted) 구조

레코드를 연결할 때, 연결할 레코드 주소를 군집하여접근 리스트(Accession List)라는 주소 어레이에 저장한다. 접근 리스트는 특정 키 값을 만족하는 모든 레코드 주소를 저장하는, 물리적으로 연속인 주소 리스트이다. 다단계 인덱스와 접근 리스트로 구성되어있으므로 1차키는 다단계 인덱스를 사용하여 접근하고, 2 차키는 접근리스트를 이용하여 접근한다.

4) Cellular 역 구조

이 구조는 연결할 레코드들이 저장된 블럭의 주소를 접근 리스트에 저장하는 역구조의 변형이다. 같은 키 값을 갖는 레코드들이 여러 블럭을 차지할 경우 평균 접근 리스트의 길이를 줄일 수 있다. 또

한블럭의 입/출력 시간을 줄이므로 질의어 수행시간을 줄이게 된다.

이상의 자료 검출 형태에 따른 자료접근 방법을 종합하면 <그림2-1>과 같다.

사용자 응용 분류	접 근 기 법
전체검출 다수검출 (10 - 100%)	순차 접근 물리적 순차 구조 연결된 순차 구조
단일검출 (한개 또는 없음)	직접 접근 임의 접근 Full Indexed(Indexed Ran om) 인덱스 순차 구조 이진 트리 B-트리 Radix 검색 트리(TRIE)
일부검출 (0 - 10%)	다중리스트 역구조 Cellular 역구조

그림 2-1 검출 형태에 따른 자료접근 기법의 분류

■. 추출 질의어의 분류

액세스 기법의 성능을 분석하기 위해서 필요한 질의어는 사용자 질의어와는 다르다. 접근 방법의 추출 질의어는 레코드 추출 조건에 의해서 분류된다. 레코드 추출 조건은 레코드의 속성(Attribute)를 단위로하며, 각 속성이 가질 수 있는 값의 범위로 구성된다.레코드의 추출 조건에 의해 추출 질의를 구분하면 다음과 같다[4,10].

1) 완전 매치 질의어 (Exact-Match Query)
: 모든 속성이 명시된 질의어이다.
2) 부분 매치 질의어 (Partial-Match Query)
: 일부 속성 값만 명시된 질의어이다.
3) 범위 질의어 (Range Query)
: 모든 속성에 유한 범위값이 명시된 질의어이다.
4) Best 매치 질의어
: 완전 또는 부분 매치 질의어에 거리 함수(Distance - function)가 명시된 질의어이다. 거리 함수를 이용하여 범위 또는 부분 범위 질의어로 변환된다.

추출 질의어에 의한 성능 특성을 비교하기 위하여 다음과 같이 기호를 정의한다.

N : 화일 내의 레코드 수
F : 범위 질의어를 만족하는 레코드 수
K : 다중 키 탐색 방법에서 키에 속한 속성 수
t : 부분매치 질의어에서 명시된 속성 수

Ⅲ. 단일키 인덱싱 기법의 비교

인덱싱(Indexing)이란 화일에서 레코드를 액세스하기 위한 자료구조의 기본 기술이다[15]. 인덱스 화일은 책에서의 차례표나 찾아보기와비교될 수 있다. 책의 인덱스(찾아보기)는 여러개의 엔트리로 구성되며 모두 한 쌍을 이룬다. 즉,

(주제어, 페이지 번호)

인덱스화일은 바로 책의 인덱스와 같다. 인덱스 화일은 대개 아래 형태의 레코드를 갖게 된다.

(키 값 , 주 화일에 대한 포인터)

본 장에서는 주 화일에 삽입,삭제가 이루어지는 동안 인덱스 구조가바뀌지 않는 정적 인덱스 기법과, 주 화일에 변화가 있을 때 인덱스구조가 적절히 조절되는 동적 인덱스 기법으로 구분하고, 그외 인덱스기법이 필요한 화일 조직을 기타로 묶어서 비교하였다.각각의 기법에 대한 논의 다음 지면을 할애 하기로 하고 단일 키에 대한 비교 분석한 결과를 다음과 같이 요약할 수 있다.

■. 단일키 인덱싱 기법의 성능 특성 비교

적절한 인덱스 조직 기법의 선택에 영향을 주는 요소는 저장 장치의 특성은 물론 그 자료에 접근하는 프로그램의 특성과도 밀접한 관계가있다. 즉 프로그램에 의해 수행 되어야 할 연산의 성질에 따라 인덱스조직 기법을 달리 하여야 한다.

정적 인덱스는 주 화일에 삽입,삭제가 이루어지는 동안 인덱스구조가 바뀌지 않으며 레코드가 순차적으로 구성되어 있는 구조이다. 따라서, 순차적인

검색이 용이하다. 정적 인덱스의 한가지 단점은 삽입이생길 때마다 전체의 오버플로우가 길어져 성능이 떨어진다.

정적 인덱스의 전체적인 성능을 회복하기 위해서는 주기적으로 유지,보수를 해야만 한다. 이 시간 동안은 화일을 액세스하지 못할 것이다.

이와는 달리 동적 구조에서는 효율성을 유지하기 위해 단계적으로 그 모양을 바꾸게 된다. 즉, 주 화일에 삽입,삭제가 이루어지는 동안 인덱스 구조가 바뀌게 된다. 정적 인덱스와 비교하여 레코드의 삽입,삭제에 더 많은 처리가 필요하지만 정기적인 유지,보수 수행을 별도로 할 필요는 없게 된다.

대부분의 동적 인덱스는 트리로 이루어져 있다. 이 장에서는 B-트리와 그의 변형된 B(*)-트리, B(+)-트리를 설명하였다. B-트리의노드는 R 개의 레코드와 R + 1개의 포인터를 갖는 구조로, 이진트리에서 소요되는 긴 검색 시간을 단축시킨다.

B-트리의 특성을 유지하기 위해 삽입,삭제시 노드의 분할(split)과합병(merging) 과정을 거치게 된다. B-트리에서의 키 값에 의한 순차탐색은 트리의 각 노드를 순회함으로써 수행되는데, 한 노드의 다음키가 액세스 되기 전에, 그 키 앞에 있는 모든 키에 관련된 서브트리를 순회해야 한다. 따라서 B-트리에서 키에 의한 순차탐색은 효율적이지 못하다.

B(*)-트리는 B-트리의 각 노드가 적어도 반을 채워야 하는데 반하여, 노드의 (2M-2)/3 +1 을 채움으로써 저장유효율을 67%로 증가시켰다. 즉, 노드의 빈번한 분할을 줄임으로써 삽입처리를 개선하였다.

그러나 B(*)-트리에서 노드 크기의 범위는 동일 차수의 B-트리에서보다 작기 때문에, 트리가 가변성이 크면 결과적으로 언더플로우나 오버플로우를 처리해야 하는 경우가 많아진다. 따라서 삽입,삭제가 빈번한 경우 B-트리 보다 평균적으로 성능이 떨어지게 된다.

B(+)-트리는 레코드들을 단노드에만 두고, 단노드들을 서로 연계함으로써 B-트리의 순차처리를 개선하였다. 즉, 순차처리와 직접수행이 모두 필요한 작업에 적합하다.

레코드들을 직접접근할 필요가 있는 경우 직접화일 구조를 이용한다.동적 해쉬 구조는 임의의 이진 스트링을 갖는 두개의 해쉬함수를 이용한다. 주소계산이 비교적 용이한 반면, 화일이 커질 경우 루트에서 리프까지의 경로 위에 있는 모든 노드들이 주기억공간에 넓게 분산되어 액세스 시간이 증가하게 된다.

확장 해쉬 기법은 디렉토리와 한 세트의 리프로 구성된다. 동적 해쉬가 두개의 함수를 이용하는 것에 비해 확장 해쉬는 단일함수를 사용한다. 확장 해쉬는 미래의 저장공간을 확보할 필요가 없기 때문에 저장공간을 절약할 수 있다.

이상에서 화일 인덱스 구조의 구조적인 특성을 살펴보았다. 추출 질의어에 의한 일반적인 성능을 비교하면 다음 표와 같다.

질의어 \ 구분	인덱스 기법		해쉬 기법
	정적 기법	동적 기법	
완전 매치 탐색 성공	O(N)	O(logN)	O(1)
완전 매치 탐색 실패	O(N)	O(logN)	O(1)
범위 질의어 탐색	O(N)	O(logN+F)	O(1)
삽 입	O(N)	O(logN)	O(1)
삭 제	O(N)	O(logN)	O(1)

< 추출 질의어에 의한 성능 특성 비교>

Ⅳ. 다중키 인덱싱 기법의 비교

키가 1개인 단일키에 의한 인덱싱 기법에서 각각의 인덱스 엔트리는 주 화일의 레코드 하나씩을 구분하고 있다. 그러나 실제의 많은 응용에서는 다중키의 검색을 필요로하고 있다. 다중키 인덱스는 특정한 속성값을 가진 모든 레코드를 위치시키는 빠른 방법을 제공하고 있다. 그러나 그 댓가를 치뤄야 한다. 인덱스들이 공간을 많이 차지해서 화일이 자주 수정된다면 2차 인덱스 수정에 시간이 많이 소요될 것이다. 이 장에서는 K-d 트리, K-d-B 트리 그리고 역 화일과 그리드화일의 세부적이 구성과 각각의 특성에 대한 성능을 비교해 본다.

■. K-d 트리

K-d 트리(K-dimensional Tree)는 기본키와 보조키에 의한 검색을 단일구조(Single structure)로 결합시킨다[5]. K-d 트리는 어떤 범위전체에 걸쳐 보조키를 갖는 모든 레코드를 검색하고자 하는 경우에 응용할 수 있다. (예를 들어 $45,000이상의 소득이 있으면서도 부양가족이 둘 이상인 사람들을 모두 찾을 때 등) 이 검토를 단순화 하기위해 d를 2차원으로 제한한다. 2-d 트리는 이진 트리 각각의 모든 레벨에서 1차원을 따라 비교하는 대신, 이진 트리의 연속적인 레벨에서 2차원 사이에 비교를 번갈아 한다는 점을 제외 하고는 이진트리와 유사하다. <그림4-1>은 신장과 체중을 키로하는 2-d 트리로 루트노드에이 현수의 2개 비교 필드를 삽입한다. 그러면,

이 현수 (175, 58)

여기서 "—"은 각 노드에 대하여 비교하기 위해 사용된 필드를 나타낸다.

이 름	신 장	체중	기 타
이 현수	175	58	
박 가원	170	62	
김 무원	180	61	
조 은희	177	64	
주 대식	162	65	
민 수진	171	56	
우 병기	165	60	
유 언태	190	88	

<그림 4-1> 2-d 트리 레코드

박 가원의 데이터를 삽입하기 위해 루트 노드의 신장과 박 가원의 170을 비교 하면 루트의 175보다 작으므로 왼쪽으로 따라 내려간다.이때 왼쪽 가지(Branch)는 영이므로 그대로 삽입이 된다.

이 현수(175, 58)
↙ 신 장
박 가원(170, 62)

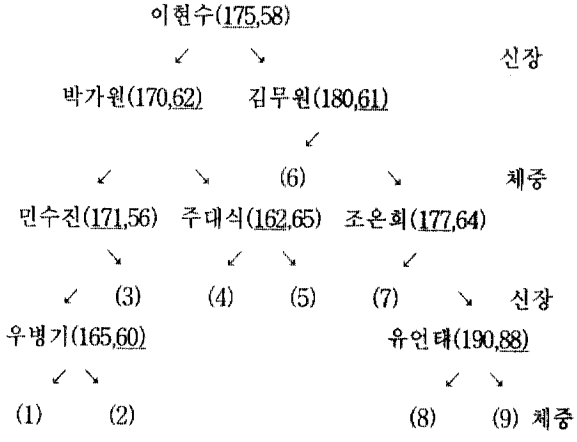
김 무원의 데이터의 삽입도 위와 같은 방법으로 비교하여 삽입한다.

이 현수(175, 58)
↙ ↘ 신 장
박 가원(170, 62) 김 무원(180, 61)

2-d 트리는 지금 2 레벨 전부가 가득 찼다. 다음 레코드 조 은희를삽입하기 위해 루트 노드의 신장과 비교한 후 레벨 1에서 체중과 비교한다. 즉, 조 은희의 신장은 루트보다 크고(177), 김 무원의 체중61보다는 크므로 김 무원의 오른쪽으로 삽입 하게된다.

이 현수(175,58)
↙ ↘ 신 장
박 가원(170,62) 김 무원(180,61)
↘ 체중
조 은희(177,64)

위와 같은 방법으로, 나머지 레코드를 차례로 삽입하면 <그림 4-2> 와같다. <그림 4-2>에 9개의 번호가 붙여져 있다. 이것은 N개 노드의 이진 트리 상태의 경우 N+1개의 영 연결(null link)을 가진 것과 같다.



<그림4-2> K-d 트리의 구조

인덱스 구성을 위해 레코드를 선정할 때 루트 레코드는 그에 대해 비교되는 차원에 대한 근사적인 중앙값을 가져야한다.그렇게 함으로써 대략 레코드들이 양분된다. 이것은 다음 레벨에서 이차원(Second Dimension)에도 적용된다. 한걸음 더 나아

가 화일과 버켓의 크기를 알면 인덱스 구성에 필요한 대표 레코드의 갯수를 추정할 수 있다.
대표 레코드 수는 2-d트리 안의 노드 수와 똑같으며 각 노드에서는 단지 필드 하나가 비교되므로 그 필드 하나만 저장하는 것이 가능하다. K-d 트리는 검색시 주로 적용되는 기본 키가 없는 그런 상황에서더욱 적합하다. 그러나, 모든 검색 필드들이 균등하게 분배되지 않을때 그 결과 인덱스 구조는 K 에 대한 값이 증가함에 따라 나빠지고 성능과 효율성도 저하된다.

■. K-d-B 트리

K-d-B 트리는 루트 노드에서 리프 노드까지의 액세스에 필요한 노드수가 모든 리프 노드에 대하여 동일하다는 점에서 항상 전체가 균형을 이루는 다중원 트리이다.

또한 B-트리의 특성과 K-d 트리의 다차원 검색 효율을 K-d-B 트리에서 예상할 수 있다.

B-트리에서와 같이 K-d-B 트리는 루트페이지의 '페이지ID'를 부여하는가변적 루트 ID와 페이지들의 집합으로 이루어져 있고, 페이지에는(영역,페이지ID)의 쌍으로 이루어진 영역 페이지(Region page)와 (점,위치)의 쌍으로 이루어진 점페이지(Point page)의 두 종류 페이지로 구성되어 있다.

이 때 "위치"는 한 데이타베이스 레코드의 위치를 부여하며, 이 (점,위치)쌍이 실제로 하나의 인덱스 레코드이다.

ROBINSON은 페이지를 여러개로 구분하고 각기 100,000만 개의 레코드 K를 2,3 차로 구분하여 질의 실험을 수행하였다[14].

실험 결과 K-d-B 트리는 다차원 동적 인덱스에 대하여 효율적인 검색 구조를 제공하며, 전 범위 질의에 대한 효율성이 매우 좋게 나타났고, 기억 장치 활용이 60% ± 10정도로 비교적 높은 효율성을 보였다.

■.역 화일

레코드들이 K개의 속성을 갖고 있는 화일에서 K가 1일 때 인덱스는 어떤 특정 속성 값을 가진 레코드를 지시한다.만일 어떤 속성 값에그러한 인덱스가 있다면 주 화일 안에 그런 속성에 대한 포인터 필드를 두어야 할 필요가 없는 것이다. K가 1인 다중 리스트 구성을 역화일이라고 한다.

실제 상업적으로 이용 할 수 있는 정보검색 시스템의 대부분이역 화일 설계를 기초로 하고 있다. 정보 검색 시스템의 사용자는 자기가 필요한 것을 주로 부울식 표현으로 나타내게 된다. 사용자와 시스템 간의 접촉(interface)은 대개가 상당히 복잡하다. 인덱스된 속성에대한 백분율로 역화(inversion)의 정도를 정의할 수 있다. 100% 역화일이라면 속성 모두가 인덱스된 것이다.

설명의 편의를 위해 <그림 4-3(a)>와 같은 자동차에 관한 정보를 포 함한 레코드 화일을 이용해 본다. <그림 4-3(b)>는 자동차 레코드 화일에 대한 역 화일의 일부를 나타낸 것이다. 주 화일은 하나의 함수로 생각할 수 있는데 즉, 레코드 주소가 주어지면 그 주소의 레코드속성 값을 돌려받는다.

화일(주소)→(속성 , 값),(속성 , 값)... ...
한 예로 ,

자동차화일(3)→(제작회사,기아),(모델 캐피탈),(색상,흑색)...

그림<4-3>의 인덱스 구조에 의해 정형화된 인덱스 화일은 역함수(inv-erse function)를 나타낸다. 그런 인덱스를 '속성-값'인 쌍으로 줄 수있고, 돌려받는 특정 속성 값을 가진 레코드의 주소 리스트를 받을 수있다.

화일(속성,값) → 주소 , 주소

자동차화일 인덱스(제작회사,기아) → 3,9,16,17,... ...이다.

주 화일이 100% 역화되었다면, 역 화일이 필요한 정보를 모두 가지고 있기 때문에 어떤 목적에 대하여는 과다하다.어느 임의의 레코드에대해서는 포인터가 너무 많을 수도 있고, 따라서 주 화일의 레코드가 이동된다면 '위치종속적'인 포인터에 많은 변경을 필요로 한다.

결과적으로 역 화일은 부울 표현식으로된 다중키 질의어에 응답하는 좋은 방법이다. 그러나 주 화일이 매우 유동성이 클 경우에는 적합치 않다. 삽입과

삭제를 위해 많은 노력이 필요하게 되며, 또한 역 리스트(Inverted List)를 보조 기억장치에서 가져와야 하는 경우, 검색효율은 더욱 떨어지게 된다.

레코드 번호	제작 회사	모델	색상	대리점
1	현 대	엑셀	백색	중구
2	대 우	코로나	적색	부구
3	기 아	캐피탈	흑색	강서
4	현 대	소나타	백색	대구
5	현 대	엑셀	청색	광주
6	대 우	르망	흑색	성동
7	현 대	엑셀	백색	서초
8	대 우	르망	청색	서대문
9	기 아	봉고	적색	명동
10	현 대	소나타	청색	속초
11	현 대	그랜저	백색	수원
12	대 우	레코드	흑색	강남
13	대 우	코로나	연두	개포
14	현 대	소나타	적색	마포
15	쌍 용	코란도	연두색	강릉
16	기 아	봉고	백색	도봉
17	기 아	캐피탈	청색	은평
18	현 대	프레스토	흑색	제주
19	현 대	엑셀	적색	안동
20	현 대	소나타	연두색	인천

(a) 자동차 레코드 화일

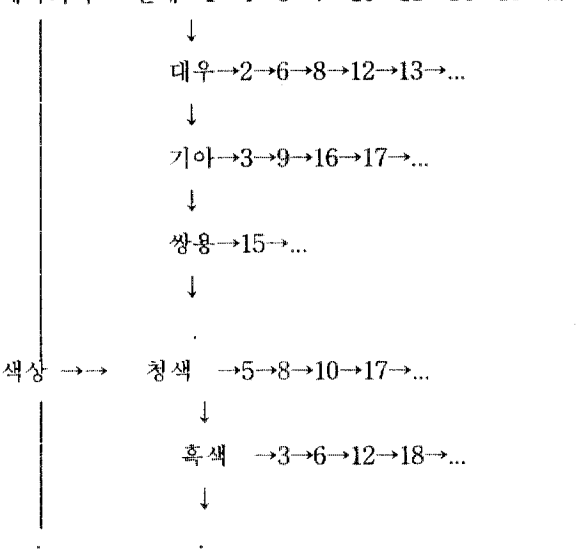
<그림4-3(a)> 역 화일의 구성

■. 그리드 화일

Nievergelt Hinterberger, Sevcik에 의해 제안된 그리드 화일은 고정된 화일이나 변동성이 큰 화일 양쪽 모두에서 기능이 좋은 보조키접근 기법이다 [12]. 설명을 위해 <그림 4-3(a)>의 자동차 화일 레코드를 사용한다.

1) 설계 목적

그리드 화일 구성의 설계 목적은 4가지로 구분된다.



(b) 역 화일의 일부

<그림4-3(b)> 역 화일의 구성

(1) 점 질의어:완전하게 서술된 질의어를 처리하는 데는 단지 두번이하의 디스크 액세스로 충분해야 한다. 점질의어 각각의 키 속성에대해서는 단일 값으로 서술되어 있다.

자동차 화일 예에서 ,

찾기: 제작회사 =대 우, 모델 =레코드, 색상 = 흑색, 대리점 = 강 남등이다.

(2) 범위 질의어:범위 질의와 부분적으로 서술된 질의어 처리도 가능해야 한다.

이런 질의어의 예로서 :

찾기: 제작회사 =대 우, 모델 = 코로나,

ㄹ< 대리점 < 밭

찾기: 제작회사 =현 대, 색상 = 녹색 등이다.

(3) 동적 적응(dynamic adaptation) : 화일 구조는 삽입 삭제가 자연스럽게 이루어져야 한다.

(4) 대칭성(Symmetry) : 기본키이건 보조키이건 모든 키 필드는 동등하게 다루어져야 한다.

2) 그리드 화일 구현의 설계

위의 4가지 설계 목적을 이루기 위해 K 차원의 비트 매트릭스(Bit-Matrix)를 생각할 수 있다. 그러나 실제로 K차원의 매트릭스는 저장하기에 너무 크다.

어떤 화일이 키가 4개이고 속성이 각각 100 개의 서로다른 값을 가지고 있다면 매트릭스는 100,000,000개의 요소(element)를 갖게 된다.

그리드 화일 구성에서는 매트릭스의 크기가 속성 값 세트를 분할 시킴으로써 줄어든다. 예로 색상을 4개의 부분 세트로 나눌 수 있다. 속성값을 문자열로 간주하면 다음과 같이 된다.

색상 < ㄹ
ㄹ≤ 색상 < ㅅ
ㅅ≤ 색상 < ㅊ
ㅊ≤ 색상

따라서 밤색은 두번째 분할에 들어간다. (ㄹ≤ 밤색 < ㅅ) 분할 포인터는 선형 범위(linear scales)에 들어있다.

각 속성에 대해 하나인 K-선형 범위의 세트는 K-차원 속성 공간위에 그리드를 정의 한다. 그래서 이 공간은 그리드 블럭들로 나누어져 있게된다. 그리드 블럭의 수는 매트릭스 요소의 수에 비해 아주 적다. 그렇지만 우리가 잃는 것은 그리드/매트릭스와 요소 간의 1대1 대응이다. 그리드 화일 구성에서 레코드들은 버켓에 저장된다. 버켓은 크기가 고정되어 있지만 화일 내에 임의대로 많이 있을 수 있다.

그리드 블럭에 대한 버켓의 동적인 할당은 그리드 디렉토리에서 관리한다. 그리드 디렉토리는 선형범위와 그리드 어레이(Grid Array)로 구성되어 있어서 각각의 요소는 버켓에 대한 포인터를 하나 갖는다.

그리드 어레이의 요소 각각이 정확히 한 버켓을 가리킨다면 K-차 직사각형을 형성하는 그리드 어레이 요소들은 동일한 버켓을 가리키게 된다. 어레이요소에 의해 지시된 버켓은 대응하는 속성 값을 가지고있는 레코드들이 저장된 화일 안에서 유일한 버켓이다.

위에서 설명한 색상 속성을 다음의 선형범위로 분할을 나타낼 수있다.

색상 [ㄹ,ㅅ,ㅊ]

다른 세가지 속성도 이같이 분할되어 있다면, 선형범위는 다음과 같이나타낼 수 있다.

제작회사 [ㄹ,ㅇ]
모델 [ㄷ,ㅂ,ㅈ,ㅌ]
대리점 [ㄷ,ㅅ,ㅊ]

다음의 레코드를 검토하면,

제작회사 = 현대, 모델 = 엑셀, 색상 = 청색,
대리점 = 수원

속성은 분할되어 각각 3,3,4 과 3로 된다. 따라서 다음의

그리드어레이 [3,3,4,3]

에 의해서 지시되는 레코드가 저장되어 있을 만한 버켓은 단 한 곳이된다. 포인터의 그리드 어레이는 정상적으로 너무커서 2차 기억장치에 보관되어야 한다.반 면에 선형범위는 정상적으로 주기억장치에 맞는다.

3) 그리드 화일의 특성

(1) 점질의어: 점질의(Point Query)에 대한 응답으로,K개의 지정된 속성값이 적절한 선형범위를 이용하여 먼저 그리드 인덱스 안으로 변형된다. K개인 인덱스 세트에 의해 그리드 어레이의 요소는 이제 디스크에서 가져올 수 있다. 직사각형의 K-차원 어레이를 선형메모리(line-ar memory) 로 사상하는 계산은 복잡하지 않다. 어느 특정 요소의주소는 계산이 간단하고 요소는 한번의 액세스로 꺼낼 수 있다.두번째 디스크 액세스로는 어레이 요소가 지시하는 버켓을 꺼낸다(fetch).

따라서 첫번째 설계 목적을 만족한다.

(2) 범위 질의어:두번째 목적을 만족시키기 위해서 그리드 어레이의임의의 축을 따라 효과적으로 이동하는 것이 가능해야 한다. 즉, 특정요소의 주소가 주어졌을 때, K차원의 어느 요소건 간에 그 다음 요소또는 그것의 앞 요소의 주소 계산이 간단해야 한다. 예를들어 다음의범위 질의어를 만족시기 위해서는

제작회사 =대 우, 모델 = 코로나, ㄹ< 대리점 < ㅌ

다음의 직사각형 내의 요소들에 의해 지시된 버켓 속의 레코드들을처리할 필요가 있다.

그리드 어레이 [1,4,1...4,2...4]

매트릭스의 연결된 리스트 수행은 이 요구사항을 충족시킬 것이다.

(3) 동적 적응성(adaption): 세번째 설계 목적은 그 구성이 삽입과삭제에 적합해야 하는 것이다.

<삽입> : 이미 다 찬 상태인 버켓에 레코드 하나를 삽입하려면 화일에 새 버켓이 할당되어서 레코드들이 두 개의 버켓으로 분할된다.

여기서 두가지 고려해야 한다. 버켓이 하나 이상의 여러개의 포인터에 의해 지시될 경우와, 버켓이 단 하나의 포인터에 의해서만 지시될경우이다. 다 찬 상태인 버켓이 그리드 어레이의 하나 이상 여러 요소에 의해 지시된다면 분할하기 위해 어떤 변화를 줄 필요는 없다. 레코드들은 현재의 분할에 의거해서 두개의 버켓 간에 분배되고 포인터의 일부가 이전 버켓에서 새 버켓으로 수정된다. 만일 그리드 요소 하나만이 버켓을 지시할 경우, 그 그리드는 수정되어야 한다. 버켓의 내용에 의해 나타나는 하부범위(Subranges)의 하나는 나누어져야 한다.

새 분할 포인트(Partition Point)가 적당한 선형범위에 추가된다. 버켓 하나에는 원래의 그리드 요소의 1/2이 할당되고 레코드들은 새로운 분할에 따라 분배된다.

<삭제> : 저장장치의 이용을 적절하게 유지하기 위해 만일 결합된 레코드의 수가 어느선 이하로 떨어진다면 2개의 버켓을 통합할 수도있다. 한쪽의 레코드들을 다른 쪽 버켓으로 옮기고 다른 쪽에 대한 포인터들을 그쪽으로 재 지정하게 된다.빈 버켓은 화일에서 재 할당된다.

(4) 대칭성(Symmetry): 이상적인 매트릭스 대칭성은 그리드 구성에서 보존된다. 기본 키와 보조 키 사이에는 성능의 차이가 없는데, 이는 인덱스된 모든 속성들이 동일한 방법으로 취급되기 때문이다.

■. 다중키 인덱싱 기법의 성능 특성 비교

이상에서 다중키 인덱싱 기법의 특성을 사례를 들어 설명하여 보았다.

다중키 인덱싱 기법은 다중키를 구성하는 모든 속성을 사용하여 탐색 성능을 향상시킨 구조이다. 다중키 구조를 K-차원의 공간인 고차공간(Hyperspace)로 간주할 경우, 다중키 방법은 고차공간을 다수의 작은 K-차원의 다면체(Hypercube)로 분할하는 방법이다. 같은 다면체에속하는 레코드들은 한 블럭 내에 위치하게 되며, K-d 트리는 기본키와보조 키 검색을 단일 구조로 결합시킨다.

K-d 트리의 검색은 이진트리 각각의 모든 레벨에서 1차원을 따라 비교하는 대신, 이진트리의 연속적인 레벨에서 K 차원 사이에 비교를 번갈아 한다는 점을 제외하고는 이진트리와 유사하다.

인덱스는 레코드들을 데이타 페이지 간에 균등하게 분배하는 것이아니므로 페이지 오버플로우 기법이 필요하게 된다. K-d트리는 검색시주로 적용되는 기본 키가 없는 상황에서 적당하다. K-d트리의 잠재적인 단점은 모든 검색 필드들이 균등하게 분배되지 않아, 그 결과 인덱스 구조는 K 에 대한 값이 증가하게 됨에 따라 나빠지게 되고, 그 성능과 효율이 저하된다.

K-d 트리의 추출질의어에 의한 성능은 다음과 같다.

1) 완전 매치 탐색	:	$O(\text{Log } N)$
2) 부분 매치 탐색	:	$O(t* N^{1-t/f})$
3) 완전 범위 탐색	:	$O(F + \text{Log } N)$
4) 부분 범위 탐색	:	$O(F + N^{1-t/k})$
5) 삽 입	:	$O(\text{Log } N)$
6) 삭 제	:	$O(\text{Log } N)$

K-d-B 트리는 K-d 트리에 B-트리의 특성을 결합함으로써 K-d 트리의 단점을 개선하였다. 즉, K-d-B 트리는 K-d 트리에서의 다차원 검색 성능과 B트리 I/O 성능의 효율성이 보장된다.

K-d-B 트리는 완전범위 질의어에 대한 효율성이 좋으나,부분 매치질의어나 부분범위 질의어에서 하나의 키로 검색될 때 K-d 트리의 검색효율이 더 좋은 것으로 나타난다.

역 화일은 속성이 1인 다중리스트로 구성되어 있