B2V-Tree: 무선 데이타 스트림에서 부분 부합 질의를 위한 색인 기법

(B2V-Tree: An Indexing Scheme for Partial Match Queries on Wireless Data Streams)

정 연 돈 † 이 지 연 ††
(Yon Dohn Chung) (Ji Yeon Lee)

요 약 이동 분산 환경에서는 무선 데이타 방송 기법을 통하여 서버의 데이타 레코드들을 이동 사용자들에게 전달하는 방식이 많이 사용된다. 그리고, 무선 방송 스트림에서 데이타를 에너지 효율적으로 접근하기 위해서는 색인 기법이 필요하다. 하지만, 기존의 색인 기법들은 데이타 레코드의 키 값을 이용한 트리 형태의 색인 구조를 사용하고 있다. 따라서 부분 부합 질의 등과 같은 내용 기반 검색 질의를 지원할 수 없었다. 본 논문에서는 무선 방송으로 이동 사용자들에게 전달되는 데이타 스트림에서 내용 기반 검색인 부분 부합 질의를 지원하기 위해 *B2V-Tree*라고 불리는 색인 기법을 제안한다. 본 논문에서 제안하는 *B2V-Tree*는 데이타 레코드들의 애트리뷰트 값을 다중 애트리뷰트 해싱을 통해 비트 벡터로 생성한 다음, 이들을 색인 트리로 구성하는 색인 기법이다.

키워드 : 부분 부합 질의, 색인, 무선 데이타 방송, 모바일 데이타베이스

Abstract In mobile distributed systems the data on the air can be accessed by a lot of mobile clients. And, we need an indexing scheme in order to energy-efficiently access the data on the wireless broadcast stream. In conventional indexing schemes, they use the values of primary key attributes and construct tree-structured index. Therefore, the conventional indexing schemes do not support content-based retrieval queries such as partial-match queries. In this paper we propose an indexing scheme, called *B2V-Tree*, which supports partial match queries on wireless broadcast data stream. For this purpose, we construct a tree-structured index which is composed of bit-vectors, where the bit-vectors are generated from data records through multi-attribute hashing.

Key words : Partial Match Query, Indexing, Wireless Data Broadcasting, Mobile Databases

1. 서 론

무선 데이타 방송이란 이동 컴퓨팅 환경에서 서버가 불특정 다수의 클라이언트들에게 방송 채널을 사용하여 데이타를 전달하는 방법이다. 서버와 클라이언트의 일대일 통신 방법에 비하여 서버로부터 전달되는 데이타를 수신만 하기 때문에 주파수 사용 및 에너지 사용의 효율성이 높은 방법이다[1,2].

무선 데이타 방송에 있어 두 가지 중요한 성능 요소로 *접근 시간(access time)*과 *튜닝 시간(tuning time)*이 있다[1,2]. *접근 시간(access time)*이란 사용자가 방송 수신을 시작한 시점으로부터 자신이 원하는 데이타를 모두 읽는 시점까지의 소요 시간을 나타내며, *튜닝 시간(tuning time)*이란 이 접근 시간 동안 실제로 방송을 들어야 하는 시간을 나타낸다. 자신이 필요로 하지 않는 데이타가 전송되는 시간 동안은 에너지 소모량이 적은 상태, 즉 *휴지 상태(doze mode)*로 지내게 된다. 그리고 실제로 방송 내용을 수신하는 동안은 *활성 상태(active mode)*로 있게 된다. 다시 말하면, 튜닝 시간은 접근 시간 동안 활성 상태로 지내는 시간을 말한다. 활성 상태에서 소모되는 에너지는 휴지 상태에서 소모되는 에너지의 수 천 배에 이른다[1]. 따라서 이동 단말기의 에너지 제약성을 고려할 때, 튜닝 시간을 단축하는 문제는 매우 중요하게 다루어지고 있다.

· 본 연구는 "정보통신기초기술연구지원사업"의 부분적인 지원을 받아 수행되었음.
† 종신회원 : 동국대학교 컴퓨터공학과 교수
ydchung@dgu.edu
†† 비 회 원 : 한국전산원 전자정부사업팀
jylee@nca.or.kr
논문접수 : 2004년 6월 25일
심사완료 : 2005년 2월 15일

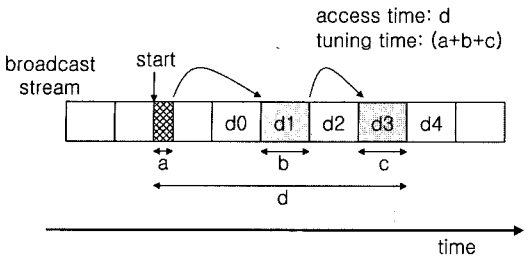

그림 1 무선 방송 데이타의 수신

그림 1에 두 데이타 개체 d1과 d3을 읽는 과정에서 사용되는 접근 시간과 튜닝 시간을 나타내고 있다. 그림에서와 같이 클라이언트가 어떤 데이타 개체를 방송 스트림에서 읽으려고 하는 경우, 해당 데이타 개체의 위치를 확인할 수 있어야 한다(방송 스트림 상에서 데이타의 위치란 그 데이타가 전달되는 시간을 의미한다). 이러한 정보-그림에서 'a' 부분에 해당-가 없다면, 클라이언트들은 자신이 원하는 데이타가 나올 때까지 방송 스트림을 계속 수신하여야만 한다. 예를 들어, 라디오에서 뉴스를 듣고자 할 때, 뉴스가 방송되는 시간을 모른다면, 뉴스가 나올 때 까지 라디오 방송을 들어야만 하는 것과 같다.

지금까지 무선 데이타 방송에 관련된 연구들에는 접근 시간을 줄이고자 하는 방법들-*데이타 캐싱(data caching)*[3,4], *불균등 데이타 방송(nonuniform data broadcasting)*[3,5], 다중 점 질의 혹은 부분 부합 질의를 위한 *클러스터링(clustering)*[6,7]-과 튜닝 시간을 줄이고자 하는 방법들-색인[1] 및 해싱[5]-이 있었다(본 논문은 튜닝 시간을 줄이기 위한 방법이다). 기존의 튜닝 시간을 줄이려는 연구들은 대부분 사용자가 데이타 레코드의 키를 사용한 *단일 점 질의(single point query)*[1]나 *다중 점 질의(multipoint query)*[6]를 사용한다는 가정을 기반으로, 디렉토리를 계층적으로 구성하는 방법들이었다.

본 논문에서는 데이타 레코드의 키를 사용하는 검색이 아니라, 여러 애트리뷰트의 값, 즉 내용을 기반으로 검색하는 *내용 기반 검색(content-based retrieval)*을 위한 색인 기법에 대하여 연구한다. 일부 혹은 전체 애트리뷰트들의 값을 기술하고, 이 값들에 부합하는 데이타 레코드들을 검색하기 때문에 *부분 부합 질의(partial-match query)*[7-9]라고 불리기도 한다. 정의 1은 본 논문에서 사용하는 내용 기반 검색 질의의 정의이다.

정의 1. 데이타 레코드가 k개의 애트리뷰트로 구성되며, 각 애트리뷰트들을 $A_1, A_2, \ldots, A_k$라고 하자. 부분 부합 질의 q는 i ($i <= k$) 개의 애트리뷰트가 특정(단일 또는 영역)값 v_j(애트리뷰트 A_j의 값)로 명세된 질의다. □

다음은 i = 3인 부분 부합 질의의 예이다.

$$q : = 'A_3 = v_3' \wedge 'A_7 = v_7' \wedge 'A_8 = v_8'$$

이와 같은 부분 부합 질의는 방송 데이타 스트림에서 자신이 원하는 데이타들이 특정 애트리뷰트의 값에만 관련되어 있는 경우에, 해당 데이타들을 효과적으로 검색해 낼 수 있는 방법이 된다. 아래의 예는 무선 정보 시스템에서 많이 서비스되고 있는 주식 정보 서비스를 바탕으로 부분 부합 질의 질의가 사용되는 모습을 설명한다.

예제 1. 이동 컴퓨팅 환경에서 이동 사용자(*mobile client*)가 방송으로 전달되는 주식 정보를 검색한다고 가정하자. 주식 정보 레코드들이 기본 키로 *종목 번호*을 가지고 있고, 그 밖에 *업종 분류 코드, 종가, 매도량, 매수량, 상/하한가, 거래량*을 애트리뷰트로 갖는다고 하자. 사용자가 기본 키로 설정되어 있는 특정 *종목 번호*의 주식에 관심을 갖고 있는 것이 아니라, *업종*이 '의약'인 주식 중에서 현재 상한가를 갖는 모든 주식에 대해 관심이 있다면 다음과 같은 부분 부합 질의를 통해 방송 데이타를 검색하게 될 것이다.

업종 = '의약' ∧ 상/하한가 = '상한가'

□

기존에 알려진 무선 데이타 스트림 색인 기법들은 레코드의 키 값을 사용하여 이루어져 있다[1,2](위의 예제를 사용한다면, *종목 번호*를 기준으로 트리 형태의 계층 구조 색인을 제공한다). 따라서, 예제 1과 같이 내용에 따른 조건을 만족하는 데이타 레코드들을 검색하는데 전혀 사용할 수 없다. 본 논문에서는 기존 연구에서 지원하지 못했던 부분 부합 질의를 위해 데이타 레코드들의 애트리뷰트 값들을 사용하여 비트벡터들로 이루어진 색인을 구성하는 방안을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 무선 데이타 스트림에서의 기존 색인 기법들에 대하여 살펴본다. 3장에서는 본 논문에서 제안하는 비트벡터를 이용한 색인의 구조와, 제안하는 색인을 사용하는 경우 이동 사용자들이 무선 방송 스트림 접근에 필요한 알고리즘을 설명한다. 4장에서는 제안하는 색인 기법이 지원하는 내용 검색 기반 질의의 유형에 대하여 기존 색인 기법들과 비교한 후, 결론 및 앞으로의 연구 방향에 대하여 기술한다.

2. 관련 연구

무선 방송 데이타 스트림의 색인이란 방송으로 전달되는 데이타들을 클라이언트들이 잘 찾을 수 있도록 하

는 시간적인 주소 정보(그림 1에서 'a'부분에 해당) 를 방송 스트림에 데이타와 함께 전달하는 방법을 말한다. 즉, 클라이언트의 튜닝 시간을 줄이고자 하는 목적이다.

지금까지 무선 데이타 방송을 위한 색인 기법으로*(1, M) Indexing* 기법, *Flexible Indexing* 기법, *Distributed Indexing* 기법 등이 제안되어 왔다[1,2]. 이 중에서 *Distributed Indexing* 기법이 가장 우수한 색인 기법으로 알려져 있다. 여기에서는 *Distributed Indexing* 기법에 대하여 살펴본다. *Distributed Indexing*은 방송 스트림으로 전달되는 데이타들에 대해 B-tree 형태의 색인을 형성한 다음, 트리의 각 노드에 해당하는 색인 정보를 데이타 방송 시 데이타들과 섞어서 방송하게 된다. 그림 2는 트리 형태로 구성된 색인 정보를 나타내며, 그림 3은 이 색인 정보들이 데이타들과 함께 방송 스트림으로 구성된 모습을 나타내고 있다. 방송으로 전송할 데이타 집합에 대한 색인 및 데이타 버켓들의 스트림으로 구성한 것을 bcast라고 부르며, 하나의 bcast 내에서는 데이타의 변경이 발생하지 않으며, bcast들 사이에서는 데이타들에 대한 변화가 발생할 수 있다. 변경된 데이타 집합에 대한 스트림은 새로운 색인 및 스트림 구성으로 bcast를 생성하여 이루어진다.

무선 방송에서는 데이타를 송수신하는 기본 단위로 *버켓(bucket)*을 사용한다[1,2]. 디스크의 *페이지(page)* 개념과 같다고 볼 수 있다. 데이타가 저장되어 있는 버켓을 *데이타 버켓*, 색인이 저장되어 있는 버켓을 *색인 버켓*이라 부른다. 편의상 색인 트리의 각 노드는 하나의 버켓으로 구성된다고 가정하자. *Distributed Indexing*은 색인 트리에서 적정 수준(level)이상의 노드들에 대해서 방송 스트림에 포함시킬 때, 중복하여 삽입한다. 이를 통해서, 이동 사용자들이 방송 스트림을 수신할 때, 자신이 원하는 데이타에 대한 위치를 보다 효과적으로 찾을 수 있게 된다. 그림 3은 그림 2의 색인 트리를 두 번째 수준의 노드들까지 중복하여 나타낸 예이다. 중복되어 방송되는 노드들을 *제어 색인 노드(control index node)*라고 하며, 각 버켓들에는(데이타 버켓 및 색인 버켓) 가장 가까운 다음 제어 색인 노드에 대한 주소 값(즉, 시간 값)이 기록된다(모든 데이타 버켓과 색인 버켓에는 가장 가까운 제어 색인의 주소 값이 기록되어 있다. 따라서, 임의의 버켓을 처음 수신하는 경우, 이 주소 값을 사용하여 제어 색인으로 이동하게 된다).

각 색인들의 내용을 살펴보자. 먼저 가장 상위 노드인 I 노드에는, a1, a2, a3 노드들에 대한 위치 값이 기록되어 있다, a1 노드에는 b1, b2, b3 노드에 대한 위치 값이 저장되며, b1 에는 c1, c2, c3, 그리고 c1에는 데이타 버켓인 d0, d1, d2에 대한 위치 값이 저장되게 된다. (그림에서 동그라미는 색인 버켓이며, 네모는 데이타 버켓이다. 논문에서는 지면이 협소한 관계로 첫 번째 데이타 버켓인 d0에 대한 그림만 나타내고 있다.) 제어 색인인 I, a1, a2, a3는 각각 자신의 자식 노드들이 나타날 때 마다, 반복하여 나타나게 된다. 가령, I 노드는 a1, a2, a3 노드가 나타날 때 마다 반복되며, a1은 b1, b2, b3가 나타날 때 마다 반복되게 된다. 즉, 트리의 차수(fan-out)만큼 반복되는 것이다. 편의상 중복되는 제어 색인들을 a1′, a1″과 같은 기호로 표기하기로 한다. a1′은 a1 색인의 첫 복사 본이며, a1″은 두 번째 복사 본을 나타낸다. a1′은 b2 색인 앞에 삽입되며, a1″은 b3 색인 앞에 삽입된다.

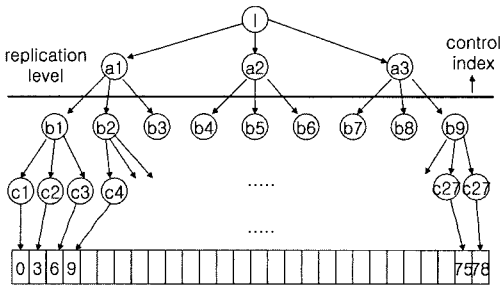

그림 2 Distributed Indexing의 색인 트리

그림 3과 같이 만들어진 방송 스트림에서 사용자들이 데이타 검색하는 과정을 예를 들어보자. 54번 데이타 버켓을 원하는 어떤 이동 사용자가 36번 데이타 버켓이 전달되는 시점에서 방송 스트림을 수신하기 시작하였다고 가정하자. 그러면, 이 사용자는 다음의 절차를 거쳐 자신이 원하는 데이타를 수신하게 된다.

① 현재 버켓(d36)을 읽고, 다음 제어 색인인 a2″의 위치를 알아낸다
② a2″ 색인 버켓이 도착할 때까지 기다린다.
③ a2″ 색인을 읽고, 다음 색인인 I″의 위치를 알아낸다.
④ I″를 읽은 후, 데이타 d54를 찾기 위한 색인들 a3, b7 그리고 c19를 차례로 따라간다.
⑤ c19 색인을 읽은 후, d54의 주소를 알아낸다.
⑥ d54가 전달되는 시점까지 기다린 후, 데이타를 읽는다.

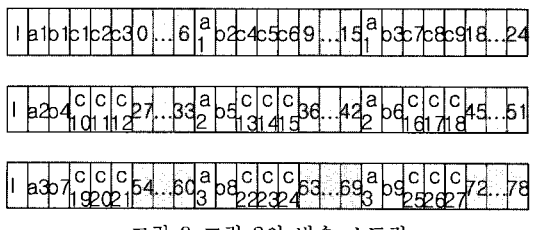

그림 3 그림 2의 방송 스트림

일반적으로, 색인 정보를 이용하면 튜닝 시간을 줄일 수 있는 반면, 데이타의 접근 시간을 증가시키는 문제점을 갖는다. 따라서, 색인 트리에서 어떤 수준의 색인 노드들까지 반복시켜서 방송 스트림을 구성하는 가의 문제는, 접근 시간과 튜닝 시간 사이의 관계와 결부되게 된다. [1]에서 이 부분에 대하여 최적의 제어 색인 수준 값을 제시하고 있다.

3. 제안하는 색인 기법

본 논문에서 제안하는 색인 기법은 데이타 레코드들을 다중 애트리뷰트 해싱(multi-attribute hashing) 기법[10]을 사용하여 *비트 벡터(bit vector)*로 나타낸 후, 비트벡터들을 계층적으로 구성하여 색인 트리를 만든다. 비트벡터들로 구성된 색인 정보는 데이타 레코드들과 함께 혼합되어 방송 스트림으로 이동 사용자들에게 전달된다.

3.1 다중 애트리뷰트 해싱을 이용한 비트 벡터 생성

다중 애트리뷰트 해싱 기법을 사용하여 데이타 레코드들을 '*데이타 비트벡터(data bit-vector)*'로, 사용자 질의를 '*질의 비트벡터(query bit-vector)*'로 표현한다. (*데이타 비트 벡터*를 *데이타 벡터*, *질의 비트 벡터*를 *질의 벡터*라고 줄여서 부르기로 한다.)

정의 2. n개의 애트리뷰트 A_1, A_2, ..., A_n으로 구성되는 데이타 레코드 집합에 대하여, 각 애트리뷰트별로 미리 정의된 해싱 함수(h)가 존재하고, h_j를 데이타 레코드 R_i의 j번째 애트리뷰트인 A_j의 값에 대한 해시 비트열(hashed bit stream)이라고 하자. 데이타 레코드 R_i에 대한 데이타 벡터 V_i는 "$h_1 \oplus h_2 \oplus ... \oplus h_n$"이다. 여기서 '⊕'는 비트열을 연결(concatenate)하는 연산자이다.

정의 3. 질의 벡터 Q_i는 질의를 규정하는 애트리뷰트들의 값들을 데이타 벡터를 생성할 때 사용한 해싱 함수를 이용하여 생성한 해시 비트 열(bit-stream)들의 결합이다(결합 순서 역시 데이타 벡터 생성에 사용된 순서와 동일하며, 질의에서 규정하지 않은 애트리뷰트에 대해서는 '*'로만 이루어진 해시 비트 열을 대신 사용한다). □

'예제 1'을 단순화 시켜 다중 애트리뷰트 해싱을 설명한다. 방송 스트림으로 전달되는 레코드들이 4개의 애트리뷰트-A_1=*종목번호*, A_2=*매도량*, A_3=*매수량*, A_4=*종가*로 구성되어 있고, 각 애트리뷰트 별로 다음과 같은 해싱 함수들이 적용된다고 가정하자.

$$h_{\text{종목번호}}(x) = \begin{cases} 00 \text{ if } x<100 \\ 01 \text{ if } 100\le x<200 \\ 10 \text{ if } 200\le x<300 \\ 11 \text{ if } 300\le x \end{cases}$$

$$h_{\text{매도량}}(x) = \begin{cases} 00 \text{ if } x<10000 \\ 01 \text{ if } 10000\le x<20000 \\ 10 \text{ if } 20000\le x<30000 \\ 11 \text{ if } 30000\le x \end{cases}$$

$$h_{\text{매수량}}(x) = \begin{cases} 00 \text{ if } x<10000 \\ 01 \text{ if } 10000\le x<20000 \\ 10 \text{ if } 20000\le x<30000 \\ 11 \text{ if } 30000\le x \end{cases}$$

$$h_{\text{종가}}(x) = \begin{cases} 000 \text{ if } x<5000 \\ 001 \text{ if } 5000\le x<10000 \\ 010 \text{ if } 10000\le x<20000 \\ 011 \text{ if } 20000\le x<30000 \\ 100 \text{ if } 30000\le x<40000 \\ 101 \text{ if } 40000\le x<50000 \\ 110 \text{ if } 50000\le x<60000 \\ 111 \text{ if } 60000\le x \end{cases}$$

레코드들은 각 애트리뷰트별로 l_i 비트만큼의 해싱된 결과들을 연결한(concatenate) $l(=l_1 + l_2 + l_3 + l_4)$ 비트 데이타 벡터로 표현된다(이 예의 경우 $l_1 = 2$, $l_2 = 2$, $l_3 = 2$ $l_4 = 3$이며, 비트 벡터의 크기 l은 9이다.) 즉, {종목번호=150, 매도량=13000, 매수량=2000, 종가=22000}인 레코드의 경우 "010100011 (= 01 ⊕ 01 ⊕ 00 ⊕ 011)"로 표현된다. (다중 애트리뷰트 해싱을 통해 생성된 비트 열들을 하나의 비트 벡터로 결합하는 순서는 애트리뷰트의 '사용 빈도' 및 '중요도' 순으로 상위(좌측)에 배치하는 방법이 최적이라고 알려져 있다[8].)

사용자의 질의도 유사한 방법으로 질의 벡터로 표현할 수 있다. 모든 애트리뷰트가 아닌 일부 애트리뷰트에 대해서만 관심을 둘 경우에 사용되는 부분 부합 질의 예로 {*매도량*≥30000, 10000≤ *종가*<20000}를 질의 벡터로 표현하면 "**11**010"이다. '*'표시는 해당 비트가 어떤 값을 갖는지를 상관하지 않는 것을 나타낸다. (비트 벡터라는 용어 자체는 '0'과 '1'만 나타낼 수 있지만, 본 논문에서는 개념 설명의 편의를 위해 '*'를 포함하도록 한다.)

3.2 BV-Tree(Bit Vector-Tree): 비트 벡터 트리

앞 절에서 설명한 다중 애트리뷰트 해싱 기법을 사용하여 무선 데이타 스트림을 구성하는 데이타 레코드들에 대한 데이타 벡터들을 구한 후, 이 데이타 벡터들에 대한 계층 구조인 *BV-Tree*를 생성한다. (본 논문은 해싱 기법을 이용한 무선 데이타 스트림 색인 기법을 제안하고 있다. 따라서, 자체적인 해싱 함수를 제안하지는 않으며, 기존의 연구에서 개발된 해시 함수를 사용한다. 해시 함수는 비트 벡터의 분포를 비교적 균등하게 생성할 수 있다고 가정한다.)

정의 4. '*l*'개의 비트로 구성되는 비트 벡터 집합에 대한 비트 벡터 트리 (*BV-Tree: Bit Vector Tree*)는 다음과 같다.

- *BV-tree*는 각 노드가 크기 '*l*'의 비트 벡터이며, 높이가 '*l+1*'인 완전 이진 트리(full binary tree)이다.
- 루트 노드는 각 비트의 값이 모두 '*'인 비트 벡터이다. (루트 노드의 깊이는 '0'으로 설정한다.)
- 깊이 '*i*'에 있는 노드는 상위 '*i*'개의 비트에 '0' 또는 '1'의 값을 갖고, 하위 '*l-i*' 개의 비트에 '*' 값을 갖는 비트 벡터 이다.
- 깊이 '*i*'에 있는 노드는 깊이 '*i+1*'에 두 개의 자식 노드를 갖는다. 이때, 자식 노드들의 비트 벡터는 부모 노드 비트 벡터의 첫 번째 '*' 비트 값을 '1'과 '0'으로 대체하고 나머지 비트 값들은 동일한 비트 벡터이다.
- 말단 노드(leaf node)는 깊이 '*l*'에 위치하며, 비트 벡터의 각 비트의 값이 '0' 또는 '1'이다. □

다음 그림 4는 4개의 비트로 구성된 비트 벡터 집합에 대한 *BV-Tree*를 나타낸 것이다. 16 (= 2^4)개의 말단 노드를 비롯하여 '****'를 루트 노드로 하는 완전 이진 트리의 형태로서, 중간(intermediate) 노드에 위치하는 비트 벡터들은 자식 비트 벡터를 포함하게 된다. '*' 표시가 '0' 또는 '1'을 나타내기 때문이다.

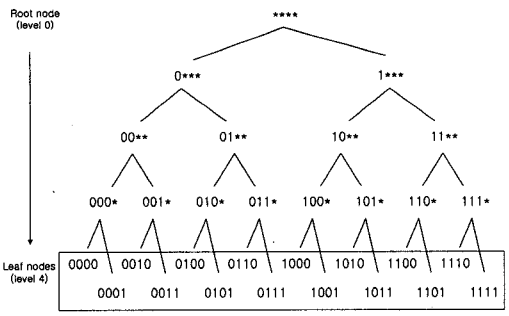

그림 4 크기 4인 비트 벡터 집합에 대한 *BV-Tree*

*BV-Tree*의 말단 노드들은 해당 비트 벡터가 의미하는 데이타 레코드들을 가리키게 되며, 말단 노드들은 다시 중간 노드들이 가리키게 되며, 최종적으로 루트 노드에 의해 색인되게 된다. 따라서, *BV-Tree* 형태의 색인 정보가 데이타 레코드들과 함께 무선 스트림으로 전송될 경우, 사용자들은 질의를 질의 벡터 형태로 나타낸 후, 스트림 상에 존재하는 *BV-Tree* 색인 노드들을 검사함으로써 찾고자 하는 데이타 레코드들의 위치를 찾아갈 수 있다.

3.3 색인 및 데이타 버켓의 구조

색인 버켓은 데이타 레코드들의 내용을 기술하는 비트 벡터들과 주소 값들의 쌍들로 구성된다. 데이타 버켓은 버켓의 대부분을 데이타 레코드들을 저장하는데 사용된다. 그리고, 모든 버켓에는 다음 색인 버켓에 대한 주소 필드와 같은 각종 제어 정보 필드를 포함한다.

정의 5. 색인 버켓을 구성하는 필드들은 다음과 같다.

- BUCKET_TYPE: 색인 버켓 혹은 데이타 버켓 여부를 나타내는 필드
- ARRAY_OF_ADDRESS_TUPLES[*M*]: *M*개의 주소 튜플들로 구성된 배열이다. 주소 튜플이란 (비트벡터, 해당 비트벡터가 가리키는 색인 및 데이타 버켓의 주소)로 구성되는 정보이다.
- LINK_TO_NEXT_INDEX: 현재 버켓에서 가장 가까운 다음 색인 버켓에 대한 주소 필드
- NEXT_BROADCAST: 다음 방송 사이클의 시작 주소

정의 6. 데이타 버켓을 구성하는 필드들은 다음과 같다.

- BUCKET_TYPE: 색인 버켓 혹은 데이타 버켓 여부를 나타내는 필드
- ARRAY_OF_DATA_RECORDS[*N*]: *N*개의 데이타 레코드들이 저장되는 배열이다.
- CONTINUE_TO_NEXT_BUCKET: 말단 비트벡터가 가리키는 데이타 레코드들이 둘 이상의 버켓에 저장되는 경우, 연속하는 데이타 버켓에도 동일한 비트 벡터에 해당하는 데이타 레코드가 존재하므로 수신하여야 함을 나타내는 부울 값을 기록하는 필드이다.
- LINK_TO_NEXT_INDEX: 현재 버켓에서 가장 가까운 다음 색인 버켓에 대한 주소 필드
- NEXT_BROADCAST: 다음 방송 사이클의 시작 주소

□

버켓의 크기와 비트 벡터의 크기, 데이타 레코드의 크기, 버켓의 주소 값을 표기하는 필드의 크기 등과 같은 시스템 환경이 결정되면, *N* 또는 *M*의 값은 다음의 수식에 따라 자동으로 결정된다.

$$N = \left\lceil \frac{(\text{the size of a bucket} - \text{the sum of all control information field sizes})}{\text{the size of a data record}} \right\rceil$$

$$M = \left\lceil \frac{(\text{the size of a bucket} - \text{the sum of all control information field sizes})}{\text{the size of an address tuple}} \right\rceil$$

3.4 B2V-Tree(Bucket-based Bit Vector-Tree): 버켓 기반 색인 트리

무선 방송으로 전송할 데이타 레코드들과 이 레코드들에 대한 해시 함수 집합이 결정되면, *BV-Tree* 형태의 비트 벡터들이 구성된다. 이들을 무선 스트림으로 전송하기 위해서는 *BV-Tree*를 색인 버켓을 기반하는 색인 트리로 구성하여야 한다.

*BV-Tree*를 색인 버켓 형태로 구성하기 위해서는 *M* (하나의 색인 버켓에 포함되는 주소 튜플의 개수 값으로 2의 지수 꼽 형태로 표현되어야 한다)개의 비트 벡터들을 하나의 노드로 통합하여 색인 트리를 구성하여야 한다. 그림 4를 예로 설명하기로 한다. 가령 *M* 값이 4인 경우 레벨 4에 존재하는 비트 벡터들을 4개씩 모아

서 하나의 색인 버켓을 구성하게 된다. 이때 하나의 색인 버켓에 모여지는 비트벡터들은 서로 인접하는 비트벡터들이 된다. 그림 5의 말단 노드들을 4개씩 묶으면 다음과 같은 4개의 말단 색인 버켓들이 만들어진다.

- 말단 색인 버켓 1 <0000, 0001, 0010, 0011>
- 말단 색인 버켓 2 <0100, 0101, 0110, 0111>
- 말단 색인 버켓 3 <1000, 1001, 1010, 1011>
- 말단 색인 버켓 4 <1100, 1101, 1110, 1111>

이들 색인 버켓들이 구성되면, 이들을 포함하는 상위 레벨의 색인 버켓이 필요하게 된다. 각 색인 버켓을 기술하는 4개의 비트 벡터들이 "00**," "01**," "10**," "11**"이므로, 이들을 통합하여 상위 레벨의 색인 버켓을 만들게 된다. 하나의 색인 버켓 추가로 모든 말단 색인 버켓을 포함하므로 더 이상의 색인 버켓은 필요하지 않게된다. 그림 5은 그림 4의 *BV-Tree*를 *B2V-Tree*로 변경한 결과를 나타낸다. *B2V-Tree*를 완성하면, 트리 검색 방법 중 깊이 우선 탐색 방법(DFS: Depth First Search)을 사용하여 트리의 색인 버켓과 데이타 버켓들 차례로 스트림으로 구성하여 생성하게 된다(본 논문에서 제안하는 *B2V-Tree* 기법은 색인 정보의 구축에 있어 비트벡터를 사용하는 것에 있기 때문에, 데이타 버켓과 색인 버켓의 배치 및 복제(replication) 등과 같은 방법은 기존 연구[1,11]에서 제안된 방법들을 별다른 수정 없이 사용할 수 있다. 그림 5에서 사용한 버켓 배치는 편의상 색인 버켓의 복제가 전혀 없는 경우로서, *Distributed Indexing*에서 제안한 제어 색인 버켓 복제를 본 *B2V-Tree*에 적용하여도 무방하다. 그림 5에서 *Distributed Indexing* 기법의 색인 복제 기법을 적용하면, 색인 제어 수준을 2로 할 때 <I_1, I_2, D_1, ... D_5, I_1',I_3, D_6, ..., D_{13}, $I_1''I_4$, D_{14}, .., D_{19}, I_1''', I_5, D_{20}, D_{21} >의 데이타 스트림이 생성된다.)

그림 5에서 음영 처리된 사각형들은 데이타 버켓을 나타내며, 그렇지 않은 사각형 들은 색인 버켓들을 나타낸다. 하나의 비트벡터에 대하여 해당 데이타 레코드들이 다수가 존재할 수 있기 때문에, 하나 이상의 데이타 버켓들이 하나의 비트 벡터에 대응되는 경우를 대비하여 정의 6에서 "CONTINUE_TO_NEXT_BUCKET" 필드를 정의했다. 그림 5에서 데이타 버켓들 사이의 링크 표시는 이 점을 나타내기 위한 표시이다. 실제로는 링크가 존재하지 않고 위 필드 값을 사용하게 된다. 또한, D_1 데이타 버켓과 같이 하나의 버켓에 서로 다른 비트벡터로 표현되는 데이타 레코드들이 함께 위치할 수도 있다. 단, 이러한 경우는 동일한 색인 버켓에 포함되어 있는 연속된 비트벡터들인 경우에 한한다. 예를 들어 "0011"로 표현되는 데이타 레코드와 "0100"로 표현되는 데이타 레코드들이 하나의 데이타 버켓이 포함될 수 있다고 하여도, 두 비트벡터들이 서로 다른 색인 버켓에 속하기 때문에, 다른 데이타 버켓들로 구분하여 전달하여야 한다. 만일 D_5 데이타 버켓에 두 비트벡터에 해당하는 데이타 레코드들이 함께 저장된다면, I_3 색인에서 D_5에 대한 주소 값을 사용할 수 없다. 시간적으로 지나간 데이타 버켓에 대한 주소 값이기 때문이다.

3.5 이동 사용자의 무선 방송 데이타 스트림 접근 방법

서버에서 B2V-Tree 색인을 사용하여 데이타 스트림

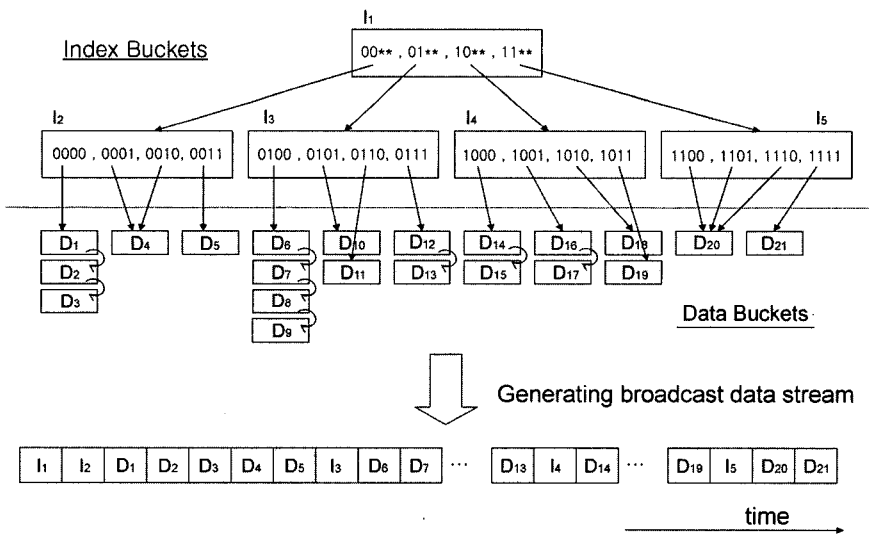

그림 5 *B2V-Tree*의 구축 및 무선 방송 데이타 스트림 생성

```
1: PROCEDURE Wireless_Data_Stream_Generation
2: INPUT: A Set of Data Records
3: OUTPUT: Broadcast Stream (Data + Index)
4:       FOR EACH data record di
5: Get hashed bit-vector bvi through given hash function H;
6: END-FOR
7: Construct BV-Tree with bvi's for all data records;
8: Add data records in the bottom-level of the BV-Tree
9: where the data records are linked to the leaf node where its hashed bit stream is stored;
10: Construct B2V-Tree through traversing BV-Tree in a BFS style where the BV-Tree nodes
11: are clustered into B2V-Tree nodes using N, M values in Section 3.3;
12: Generate Stream by traversing the B2V-Tree in a DFS style;
13: END_OF_PROCEDURE
```

그림 6 제안하는 색인 기법을 사용한 무선 데이타 스트림의 생성 알고리즘

을 생성하여 방송하는 경우, 이동 사용자들이 무선 스트림을 접근하는 프로토콜을 필요로 한다. 이동 사용자가 검색하고자 하는 데이타들에 대한 규정(specification)을 질의 벡터 형태로 표현한 다음, 사용자 질의 벡터와 스트림 상에 존재하는 색인 버켓 내의 비트 벡터들과의 관계에 따라 해당 색인 버켓들을 선택적으로 수신하여 검색 대상 데이타 버켓들에 접근하게 된다.

임의의 비트벡터 V1과 V2는 다음의 4가지 중 하나의 관계를 지닌다. 이 관계에 따라 그림 7의 접근 프로토콜에서 색인 버켓의 내용에 따라 적절한 다른 색인 버켓으로 이동하면서 데이타 버켓을 찾게 된다.

- *포함(included-by) 관계*: V1이 나타내는 데이타 레코드들의 집합이 V2가 나타내는 데이타 레코드들의 집합을 포함하는 경우 V1이 V2를 포함한다고 한다. (반대 방향의 포함관계도 가능.)
- *중첩(overlapped-with) 관계*: V1이 나타내는 데이타 레코드들의 집합과 V2가 나타내는 데이타 레코드들의 집합의 교집합이 공집합이 아닌 경우, V1과 V2는 중첩 관계에 있다.
- *서로 소(disjoint-with) 관계*: V1과 V2의 데이타 레코드 집합들에 대한 교집합이 공집합인 경우, V1과 V2는 서로 소 관계이다.
- *일치(exactly-matched-with) 관계*: V1과 V2가 동일한 비트벡터인 경우 일치 관계이다.

그림 7에 기술된 스트림 데이타 접근 프로토콜을 사용하는 몇 가지 예를 보기로 하자. 그림 5와 같이 무선 데이타 스트림이 구성되어 있을 때, (1) 이동 사용자가 질의를 시작하는 위치가 데이타 버켓 D_5이며, 이때 질의 벡터가 "0111"인 경우, (2) 시작 위치가 색인 버켓 I_1이며 질의 벡터가 "101*" 인 경우, (3) 시작 위치가 색인 버켓 I_3이며 질의 벡터가 "****"인 경우 데이타 레코드들을 검색하는 과정은 그림 8과 같다. 수평 방향 화살표는 버켓들을 순차적으로 쫓아서 읽는 것을 나타내며, 수직 방향 화살표는 그림 7의 15행에 기술된 병렬적으로 수행되는 부분을 나타낸 것이다.

(1)의 경우 현재 버켓인 D5를 수신한 다음, 가장 가까운 위치의 색인 버켓인 I3으로 이동한다 (알고리즘의 7-8행). I3 색인 버켓의 색인 비트 벡터 값을 읽고, 질의 벡터인 "0111"과 일치하는 비트 벡터가 있으므로, 이들이 가리키는 데이타 버켓인 D12, D13을 차례로 읽으면 질의가 찾고자 하는 데이타를 검색할 수 있다 (알고리즘 12-13 행).

(2)의 경우는 현재 버켓인 I1에 질의 벡터인 "101*"에 포함되는 비트벡터가 두 개 ("1010", "1011") 있음을 알 수 있다. 이들 벡터가 가리키는 데이타 버켓 주소(D18, D19)를 사용하여 데이타를 수신하면(알고리즘 14-17행), 질의 처리가 완료된다.

(3)의 경우에 대해 알아보자. I_3 색인 버켓에서 "****" 질의 벡터를 검색하고자 하는데, 현재 색인 버켓에서 지니고 있는 "0100", "0101", "0110" 그리고 "0111"의 비트벡터들이 모두 질의 벡터에 포함(included-by)되기 때문에, 현재 색인 버켓이 가리키는 모든 하부 포인터들을 쫓아가는 과정이 필요하다(알고리즘 14-17행). 추가로 현재 색인 버켓에서 커버하지 않는 나머지 데이타들을 검색하기(알고리즘 18-21행) 위해 I_4, I_5 및 I_2 색인 버켓에 대해서 색인 버켓과 데이타 버켓들을 검색하여야 한다. I_4, I_5, I_2 색인 버켓에도 질의 벡터 "****" 에 해당하는 비트벡터들이 존재하기 때문이다. (3)에서 I_1 과 I_2, 색인 버켓은 다음 방송 스트림에 나타나는 버켓들이다. (참고로, (3)의 경우 [11]에서 제안한 *SL(Sibling Link)* 혹은 *NL(Nephew Link)* 제어 색인 반복 기법을 적용할 경우, I_1 색인 버켓을 수신하

```
1: PROCEDURE Wireless_Data_Stream_Access
2: INPUT: A Query Bit-Vector (QV)
3: OUTPUT: Data Records for the Query (Result)
4: LOCAL VARIABLES: Result, BV, Temp
5:      Set Result empty;
6:      Temp ← Read the current bucket;
7:      IF (Temp is not an index bucket) THEN // Using BUCKET_TYPE field
8:              Probe to the next index bucket; // Using LINK_TO_NEXT_INDEX field
9:      ENDIF
10: LABEL START_INDEX_PROBE:
11:     read the address tuples in the current index bucket;
12:     IF (there is one bit-vector BV exactly matched with QV) THEN
13:             Result ← Result + Data_Retriever(BV.address);
14:     ELSEIF (there is one or more bit-vectors BV included by QV) THEN
15:             FOR EACH BV included by QV DO in parallel
16:                     Result ← Result + Data_Retriever(BV.address);
17:             END-FOR
18:             IF (there is more data to retrieve) THEN
19:                     Probe to the next index bucket; // Using LINK_TO_NEXT_INDEX field
20:                     GOTO START_INDEX_PROBE;
21:             ENDIF
22:     ELSEIF (there is a bit-vector BV including QV) THEN
23:             Probe to BV.address;
24:             GOTO START_INDEX_PROBE;
25:     ELSE // All bit-vectors are independent.
26:             Probe to the next index bucket; // Using LINK_TO_NEXT_INDEX field
27:             GOTO START_INDEX_PROBE;
28:     ENDIF
29:     RETURN Result to User; // Now, Result is the answer for the query
30: END_OF_PROCEDURE
31: FUNCTION Data_Retriever
32: INPUT: Address (addr)
33: OUTPUT: Data Records (Result)
34: LOCAL VARIABLES: Result, BV, Temp
35:     Set Result empty;
36:     IF (addr is previously accessed) THEN
37:             BREAK;
38:     ELSE
39:             Temp ← Read the bucket pointed by the addr;
40:             IF (Temp is a data bucket) THEN
41:                     Result ← Result + Temp ;
42:                     WHILE (Temp .CONTINUE_TO_NEXT_BUCKET is TRUE) DO
43:                             Temp ← Read the next data bucket;
44:                             Result ← Result + Temp ;
45:                     ENDWHILE
46:             ELSE    // Temp is an index bucket.
47:                     FOR EACH bit-vectors BV in the current index bucket DO
48:                             Result ← Result + Data_Retriever(BV.address);
49:                     ENDFOR
50:             ENDIF
51:     ENDIF
52:     RETURN Result;
53: END_OF_FUNCTION
```

그림 7 제안하는 색인 기법을 사용한 무선 데이타 스트림의 접근 프로토콜

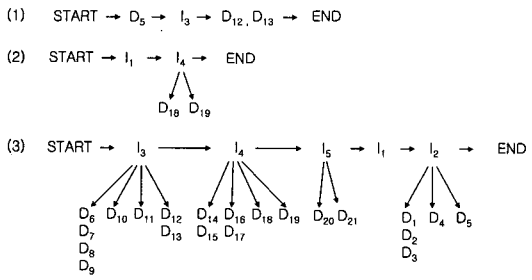

그림 8 무선 스트림 데이타 접근 프로토콜 사용 예제

지 않고 I_2 색인 버켓으로 직접 이동하는 것이 가능하다. 제안하는 방법은 색인 정보의 내용과 그 구성에 초점을 두고 있기 때문에, 색인 버켓의 배치 및 복제에 대한 방법은 기존 연구 기법들을 모두 활용할 수 있다.)

4. 평 가

본 논문에서 제안한 B2V-Tree 색인 기법은 무선 방송 데이타 스트림에서 내용 기반 검색의 하나인 부분 부합 질의 처리가 가능하도록 지원하는 방법이다. 지금까지 이 분야에서 언급되어온 색인 기법들은 데이타 레코드들의 키(primary key) 값을 기준으로 구축되어 왔다. 따라서 본 논문이 대상으로 하는 부분 부합 질의는 전혀 지원하지 못하는 실정이다(키 값을 기준으로 작성된 색인을 사용하는 무선 데이타 스트림에서 부분 부합 질의를 처리하려면, 무선 스트림상의 모든 데이타 버켓을 읽고, 내용을 보면서 검색 질의 요건에 부합하는지 확인하여야 한다).

4.1 튜닝 시간 평가

기존의 색인 기법인 Distributed Indexing을 사용하는 경우 단일 키를 사용하여 점 질의를 수행할 경우, "1 + Distributed Indexing Tree의 높이 + 해당 데이타 버켓의 개수" 만큼의 버켓 수신량을 필요로 한다. 처음에 나타난 '1' 값은 방송을 수신한 다음, 다음 방송의 시작으로 이동하기 때문에 반드시 하나의 버켓은 수신하여야 하는 값을 나타낸다. 여기서 색인 트리의 높이는 버켓의 크기와 트리의 차수(fan-out 또는 degree)가 주어지면 계산 가능한 값이다. 해당 데이타 버켓의 개수는 질의에서 주어진 키 값을 갖는 레코드들의 개수와 레코드의 크기를 곱한 값을 버켓의 크기로 나눈 값이 된다 (3.3 절에서 N 값을 계산한 방식의 역으로 생각할 수 있다).

반면에, 제안하는 방법을 사용하여 비트 벡터 형태로 질의가 주어질 경우, 버켓 수신량은 리프 노드의 비트 벡터까지 탐색할 경우, "1 + B2V-Tree의 높이 + 해당 데이타 버켓의 개수"가 된다. 해당 데이타 버켓이란 질의에서 주어진 비트 벡터에 해당하는 (3.5 절에서 설명한 '일치' 및 '포함' 개념에 해당한다.) 데이타 레코드들을 저장하고 있는 버켓이다. B2V-Tree의 높이는 데이타 레코드들의 표현하는 비트 벡터의 크기와 서로 다른 비트 벡터들의 개수에 따라 계산된다. 또한, 키를 기준으로 점 질의한 경우와 달리, B2V-Tree 중간 노드에 해당하는 비트 벡터를 검색하는 경우에는 트리 높이 만큼의 버켓 수신이 필요하지 않다.

기존 방법은 키 애트리뷰트 값을 기준으로 색인 트리를 생성하므로, 데이타 레코드의 개수에 따라 색인 트리의 모양이 결정되지만, 제안하는 방법은 레코드가 포함하는 모든 (경우에 따라 일부의 애트리뷰트만 사용할 수도 있음) 애트리뷰트가 나타내는 데이타 공간 (data space)를 분할하는 색인 트리를 만들게 되므로, 스트림으로 전송할 데이타 내용에 따라 색인 트리의 크기가 달라진다. 기존 방법의 색인 트리와 제안하는 방법의 색인 트리의 크기는 다음과 같이 계산된다.

3.3절에서 설명한 색인 트리의 차수(M)을 구하는 다음의 식에 따라, 먼저 색인 튜플(address tuple)의 크기를 결정하여야 한다.

$$M = \left\lfloor \frac{(\text{the size of a bucket} - \text{the sum of all control information field sizes})}{\text{the size of an address tuple}} \right\rfloor$$

- 기존방법: 색인 튜플의 크기 = "키 애트리뷰트의 크기 + 주소 필드의 크기"
- 제안하는 방법: 색인 튜플의 크기 = "비트 벡터의 크기 + 주소 필드의 크기"

색인 트리의 리프 노드의 개수는 기존 색인의 경우, 데이타 레코드의 개수에 따라 정해진다. 따라서, 이를 위에서 구한 M 값으로 클러스터링 시키면 버켓 단위로 구성된 색인 트리의 리프 노드들의 개수가 결정되며, 이를 균형 트리로 만들게 되므로, 다음과 같이 트리의 높이도 계산 가능하다. 아래 식에서 W는 리프 버켓의 개수를 나타낸다. 기존 색인의 경우 "⌊ 데이타 레코드의 개수/M⌋ "이며, 제안하는 색인의 경우 "⌊ 서로 다른 비트 벡터의 개수/M⌋ "이다.

$$Height\ of \in dex\ Tree = \log_M W + 1$$

위에서 계산한 버켓 수신량은 Distributed Indexing 방법과 B2V-Tree를 통한 기본적인 질의에 대한 값으로, 사용자의 질의 유형에 따라 "해당 데이타 버켓"의 개수가 달라질 수 있다. 또한, 부분 부합 질의를 사용하는 경우, 이를 여러 개의 '단일 키 점 질의'로 분해하여 수신할 경우, 위에서 계산한 식에 분해된 점 질의들의 수를 곱해야 하는 경우도 생긴다.

4.2 접근 시간 평가

색인의 사용은 튜닝 시간의 감소를 가져오지만, 접근 시간의 경우 오히려 증가하게 된다. 질의의 접근 시간을

(1) 대상 데이타 레코드를 찾는 과정과 (2) 대상 데이타 레코드를 찾은 후 다운로드 하는 과정으로 구분하여 기술하면, 대상 데이타를 찾는 과정에 소요되는 접근 시간은 확률적으로 전체 방송 스트림 크기의 절반이 되므로, "0.5 * (색인의 양 + 데이타 레코드 집합의 크기)"가 된다. 레코드 집합의 크기는 제안하는 방법과 기존 방법 모두 동일하므로, 색인의 양 차이가 접근 시간의 차이를 나타내게 된다. 기존 방법과 제안하는 방법의 색인 구성의 차이는 단일 키 값들을 사용한 트리와 해시 비트 벡터를 사용한 트리의 차이이다. 4.1절에서 계산한 색인 트리의 높이와 버켓의 크기로부터 각 색인의 양을 다음과 같이 계산할 수 있다. (H_{DI}는 는 기존 색인 트리의 높이 이며, H_{B2V}는 제안하는 B2V-Tree의 높이이다.)

$$Amount\ of \in dex_{\ Distributed \in dexing} = \frac{M^{H_{DI}}-1}{M-1} * BucketSize$$

$$Amount\ of \in dex_{\ B2V-Tree} = \frac{M^{H_{B2V}}-1}{M-1} * BucketSize$$

4.3 예제를 통한 비교

다음의 표 1은 본 논문에서 제안한 B2V-Tree 방법과 기존의 Distributed Indexing 방법을 사용하는 경우에 처리할 수 있는 질의 유형에 대하여 비교 평가하고 있다. 기존의 색인 기법을 사용하는 경우에는 부분 부합 질의에 대하여 모든 데이타 레코드들을 수신한 다음 내용을 확인하여야하기 때문에, 성능의 관점에서는 본 논문에서 제시한 방법과 직접적인 비교를 할 수 없다. 하지만, 부분 부합 질의를 위한 색인의 필요성을 보이는 목적으로 그림 5에서 사용한 간단한 예제 데이타 집합을 사용하여 제안하는 방법과 기존 색인 방법에서 필요로 하는 데이타 버켓의 수신량을 정량적으로 비교하여 표 2에 기술하였다(실제 데이타 집합이 커 지는 경우 이 차이는 더욱 증가하게 되며, 이 표에 기술한 수치는 단순히, 독자의 이해를 돕기 위한 것이다.) "수신 버켓의 개수 * 버켓의 크기 / 무선 통신의 전송 속도"의 식을 이용하며, 실제 이동 사용자가 질의 처리에 필요한 튜닝 시간을 계산할 수 있다.

5. 결 론

본 논문에서는 무선 데이타 스트림에서 부분 부합 질의를 효과적으로 지원하기 위한 색인 기법으로 B2V-Tree 색인 구조를 제시하였다. 기존 연구들은 키 애트리뷰트를 기준으로 한 탐색 트리(search tree) 형태로 내용 기반 검색 질의인 부분 부합 질의를 효과적으로 처리할 수 없었다.

제안하는 방법은, 스트림으로 전송될 데이타 레코드들의 각 애트리뷰트 값을 통해, 해시 비트열을 생성하고,

표 1 제안하는 색인 기법과 기존 색인 기법이 지원하는 질의의 유형 비교

질의 유형	B2V-Tree	Distributed Indexing
키 애트리뷰트를 사용한 점 질의 (point query with primary key)	- 키 애트리뷰트에 대한 해시 함수를 키 값의 크기만큼 할당하면 효과적인 지원이 가능 - 키 애트리뷰트에 대한 특별 처리가 없는 경우, 대상 데이타레코드가 포함되는 비트벡터가 가리키는 다수의 데이타 버켓을 검색할 수 있음 (다소 비효율적임.)	- 효과적으로 지원
키 애트리뷰트를 사용하지 않은 점 질의 (point query without primary key)	- 해당 애트리뷰트 값에 대한 비트 벡터 정보 검색을 통해 검색 대상 데이타 버켓의 수를 경감 시킬 수 있음.	- 처리하지 못함 - 모든 데이타 버켓을 수신하여야 함.
키 애트리뷰트를 포함한 부분 부합 질의 (partial-match query with primary key)	- 비트 벡터 형태로 표현되는 부분 부합 질의의 경우 최적의 데이타 버켓만을 검색함. - 데이타 버켓의 검색량을 현저히 줄임.	- 키 애트리뷰트 값만 검색에 활용 가능 - 다른 애트리뷰트에 대한 검색 조건은 데이타 버켓 검색에 사용되지 못함.
키 애트리뷰트를 포함하지 않는 부분 부합 질의 (partial-match query without primary key)	- 비트 벡터 형태로 표현되는 부분 부합 질의의 경우 최적의 데이타 버켓만을 검색함. - 데이타 버켓의 검색량을 현저히 줄임.	- 처리하지 못함 - 모든 데이타 버켓을 수신하여야 함.
키 애트리뷰트를 포함한 영역 질의 (range query with primary key)	- 질의의 각 애트리뷰트별 영역 조건을 비트 벡터 형태로 변화하여 효율적인 검색 가능 - 데이타 버켓의 검색량을 현저히 줄임.	- 키 애트리뷰트 값으로 지정된 조건만 색인을 통해 사용되며, 다른 애트리뷰트의 검색 조건은 사용되지 못 함. - 많은 데이타 버켓들을 검색하여야 함.
키 애트리뷰트를 포함하지 않는 영역 질의 (range query without primary key)	- 질의의 각 애트리뷰트별 영역 조건을 비트 벡터 형태로 변화하여 효율적인 검색 가능 - 데이타 버켓의 검색량을 현저히 줄임.	- 처리하지 못함 - 모든 데이타 버켓을 수신하여야 함.

표 2 제안하는 색인 기법과 기존 색인 기법 데이타 버켓 수신량 비교

질의 유형	*B2V-Tree*	*Distributed Indexing*
키 애트리뷰트를 사용한 점 질의 (point query with primary key)	- 평균 1, 최대 4, 최소 1개의 데이타 버켓을 수신[1)] - 키 애트리뷰트에 특별한 해시 함수를 적용할 경우 기존 방법과 동일함.	- 평균, 최대, 최소 모두 1개의 데이타 버켓 수신
키 애트리뷰트를 사용하지 않은 점 질의 (point query without primary key)	- 평균 1, 최대 4, 최소 1개의 데이타 버켓을 수신	- 21개 데이타 버켓 모두 수신
키 애트리뷰트를 포함한 부분 부합 질의 (partial-match query with primary key)	- I_2~I_5의 말단 노드 비트 벡터로 표현되는 질의의 경우 평균 1, 최대 4, 최소 1개의 데이타 버켓을 수신 - I_1 수준의 비트 벡터로 표현되는 질의의 경우 평균 5.6, 최대 10, 최소 2개의 데이타 버켓 수신[2)]	- 키 애트리뷰트에 대하여 지정된 값을 갖는 모든 데이타 버켓 수신 - 최소 1, 최대 21개의 데이타 버켓 수신[3)]
키 애트리뷰트를 포함하지 않는 부분 부합 질의 (partial-match query without primary key)	- 위와 동일	- 21개 데이타 버켓 모두 수신
키 애트리뷰트를 포함한 영역 질의 (range query with primary key)	- 질의의 각 애트리뷰트별 영역 조건을 비트 벡터 형태로 변환 가능한 경우 위와 동일	- 키 애트리뷰트에 대하여 지정된 값을 갖는 모든 데이타 버켓 수신 - 최소 1, 최대 21개의 데이타 버켓 수신
키 애트리뷰트를 포함하지 않는 영역 질의 (range query without primary key)	- 위와 동일	- 21개 데이타 버켓 모두 수신

이들을 연결하여 레코드들에 대한 비트 벡터를 생성하였다. 이 비트 벡터들은 비트 벡터들 사이의 포함관계에 따라 트리 형태의 관계를 생성하게 되는데, 이를 BV-Tree라고 한다. BV-Tree를 무선 통신에서 사용하는 버켓 단위로 클러스터링 한 결과를 B2V-Tree라고 부른다. B2V-Tree는 버켓 단위로 구성된 비트 벡터들의 탐색 트리이다.

제안하는 방법은 데이타 레코드들의 내용을 기반으로 색인을 구성하기 때문에, 영역 질의 및 부분 부합 질의에 효과적이며, 비트 벡터를 통해 데이타 공간을 분할하기 때문에, 데이타 레코드의 개수와 무관한 특징을 갖는다. 하지만, 해시 함수를 사용하여 각 애트리뷰트들의 비트 벡터를 결정하기 때문에, 오히려 키 값을 사용하여 검색하는 질의의 경우 본 논문에서 제안하는 방법이 비효과적인 경우가 생길 수 있다. 하지만, 비트 벡터의 생성 시 키 애트리뷰트에 대해서는 키 값 전부를 비트벡터로 표시하도록 데이타 벡터 및 질의 벡터를 생성하는 방법을 쓰면, 키 값을 사용하는 질의에도 사용 가능하다.

향후 연구 과제로는 [4]와 같은 캐시 제어 데이타 방송에 본 색인 기법을 적용하는 방안과, 불균등 데이타 방송 환경[3]에서 데이타 뿐만 아니라 색인 정보까지 불균등하게 구성하는 방법에 대한 연구가 필요하다. 또한, 스트림 데이타 클러스터링 기법[6, 7]을 사용하는 환경에서의 색인 방법에 대해서도 관심이 필요할 것으로 판단된다.

1) 평균은 (3+1+1+4+1+1+2+2+2+1+1+1+1)/16로 계산되며, 최대는 "0100"에 해당하는 데이타 레코드를 검색하는 경우, 최소는 "1100"에 해당하는 데이타 레코드들 검색하는 경우이다.

2) 평균은 (5+10+6+2)/4로 계산되며, 최대는 "01**"에 해당하는 부분 부합 질의를 검색하는 경우이며, 최소는 "11**"에 해당하는 질의를 검색하는 경우이다.

3) 질의에 포함된 키 애트리뷰트의 값을 갖는 데이타 버켓이 1개 뿐인 경우가 최소에 해당하며, 키 값을 갖는 데이타가 모든 데이타 버켓에 산재한 경우가 최대에 해당한다.

참 고 문 헌

[1] T. Imielinski, S. Viswanathan, and B. R. Badrinath. "Energy-efficient Indexing on Air," In *Proceedings of ACM SIGMOD Conference*, pp. 25-36, 1994.

[2] T. Imielinski and B. R. Badrinath. "Data Management for Mobile Computing," *SIGMOD RECORD*, Vol. 22, No. 1, pp. 34-39, 1993.

[3] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. "Broadcast Disks : Data Management for Asymmetric Communication Environments," In *Proceedings of ACM SIGMOD Conference*, pp. 199-210, 1995.

[4] J. Jing, O. Bakhres, A. Elmagarmid, and R. Alonso. "Bit-Sequences : A Adaptive Cache Inva-

lidation Method in Mobile Client/Server Environments," *Mobile Networks and Applications (MONET)*, Vol. 2, No. 2, pp. 115-127, 1997.
[5] T. Imielinski, S. Viswanathan, and B. R. Badrinath. "Power Efficient Filtering of Data on Air," In *Proceedings of Extending Database Technology*, pp. 245-258, 1994.
[6] Y. D. Chung and M. H. Kim. "Effective Data Placement for Wireless Broadcast," *Distributed and Parallel Databases*, Vol. 9, pp. 133-150, 2001.
[7] J. Y. Lee, Y. D. Chung, Y. J. Lee and M. H. Kim. "Gray Code Clustering of Wireless Data for Partial Match Queries," *Journal of Systems Architecture*, Vol. 47, pp. 445-458, 2001.
[8] K.A.S. Abdel-Ghaffar and A. E. Abbadi. "Optimal Disk Allocation for Partial Match Queries," *ACM Transactions on Database Systems*, Vol. 18, No. 1, pp. 132-156, 1993.
[9] M. H. Kim and S. Pramanik. "Optimal File Distribution for Partial Match Retrieval," In *Proceeedings of ACM SIGMOD Conference*, pp. 173-182, 1988.
[10] C. Faloutsos. "Multiattribute Hashing Using Gray Codes," In *Proceedings of ACM SIGMOD Conference*, pp. 227-238, 1986.
[11] Y. D. Chung and M. H. Kim, "An Index Replication Scheme for Wireless Data Broadcasting," *Journal of Systems and Software*, Vol. 51, No. 3, pp. 191-199, May, 2000.

정 연 돈
정보과학회논문지 : 데이타베이스
제 32 권 제 2 호 참조

이 지 연
1993년 8월 동국대학교 전자계산학과 학사. 1996년 2월 한국과학기술원 전산학과 석사. 2001년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 2001년 3월~2004년 1월 현대정보기술 정보기술연구소 책임연구원. 2004년 2월~현재 한국전산원 전자정부사업팀 선임연구원. 관심분야는 E-Government, Database Systems, Bioinformatics, Mobile Computing 등