범위 모자이크 질의의 효율적인 수행
(Efficient Execution of Range Mosaic Queries)

홍 석 진 † 배 진 욱 † 이 석 호 ††
(Seokjin Hong) (Jinuk Bae) (Sukho Lee)

요 약 질의 영역에 대한 단일 값의 통계 정보를 반환하는 범위 집계 질의와는 달리, 범위 모자이크 질의는 질의 영역 내의 데이타 분포를 모자이크 형태로 반환한다. 즉, 범위 모자이크 질의는 질의 영역을 다차원 격자로 나눈 후, 나뉜 각 영역에 대해 집계값을 구해서 결과로 반환하는 질의이다. 이 논문에서는 범위 모자이크 질의와, 범위 모자이크 질의를 SQL문으로 표현하기 위한 mosaic-by 연산자를 제안한다. 그리고 이 논문에서는 집계 R-트리를 이용한 범위 모자이크 질의의 효율적인 수행 알고리즘을 소개한다. 알고리즘은 모든 모자이크 셀의 집계값을 한 번의 트리 순회만으로 계산하며, 집계 R-트리의 집계값을 이용하여 질의 영역 내의 모든 노드를 접근하지 않고도 작은 수의 노드 접근만으로 질의를 수행할 수 있다. 실험 결과를 통해 제안된 알고리즘이 생성된 데이타와 실제 데이타 모두에 대해 좋은 성능을 보이는 것을 알 수 있다.

키워드 : 범위 모자이크 질의, mosaic-by, 집계 R-트리

Abstract A range mosaic query returns distribution of data within the query region as a pattern of mosaic, whereas a range aggregate query returns a single aggregate value of data within the query region. The range mosaic query divides a query region by a multi-dimensional grid, and calculates aggregate values of grid cells. In this paper, we propose a new type of query, range mosaic query and a new operator, mosaic-by, with which the range mosaic queries can be represented. In addition, we suggest efficient algorithms for processing range mosaic queries using an aggregate R-tree. The algorithm that we present computes aggregate results of every mosaic grid cell by one time traversal of the aggregate R-tree, and efficiently executes the queries with only a small number of node accesses by using the aggregate values of the aggregate R-tree. Our experimental study shows that the range mosaic query algorithm is reliable in terms of performance for several synthetic datasets and a real-world dataset.

Key words : range mosaic query, mosaic-by, aggregate R-tree

1. 서 론

범위 집계 질의(range aggregate query)는 특정 질의 영역 내의 데이타 요소에 대한 합, 평균 등의 집계값을 반환하는 질의로, 범위 통계 질의(range statistical query)의 한 종류이다. 이러한 범위 집계 질의는 공간 데이타베이스[1]와 데이타 웨어하우스[2] 환경에서 분석을 위한 유용한 도구로 사용된다. 하지만 범위 집계 질의는 질의 영역에 대한 단일 집계값 만을 반환하기 때문에, 질의 영역 내의 데이타의 분포를 알아내기에는 적합하지 않다. 이 논문에서는 질의 영역 내의 데이타에 대한 개략적인 분포를 알아내는 질의로 범위 모자이크 질의(range mosaic query)를 제안한다. 범위 모자이크 질의는 질의 영역을 격자 형태의 모자이크로 나눈 후, 나뉜 각 영역에 대한 집계값을 구하는 질의이다.

범위 모자이크 질의는 다양한 분야에 적용할 수 있다. 움직이는 자동차의 위치를 저장하는 시공간 데이타베이스에서 특정 시간 구간 동안 주어진 영역 내에 있는 차량의 대략적인 분포를 구하는 질의는 대표적인 범위 모자이크 질의라 할 수 있다. 질의 영역을 시공간의 3차원 격자로 나누고, 나뉜 각 영역에 대해 count 연산을 수행하는 범위 모자이크 질의를 통해, 특정 시간 구간 동

· 본 연구는 2005년도 두뇌한국21사업과, 정보통신부의 대학 IT연구센터(ITRC) 지원을 받아 수행되었습니다.

† 학생회원 : 서울대학교 컴퓨터공학과
jinny@db.snu.ac.kr
jinuk@db.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
shlee@cse.snu.ac.kr

논문접수 : 2005년 1월 24일
심사완료 : 2005년 7월 29일

안 주어진 영역 내의 차량 분포를 구할 수 있다. 범위 모자이크는 데이타 웨어하우스 환경에도 적용 가능하다. 예를 들어 고객의 수입과 나이를 차원 애트리뷰트로, 판매량을 값 애트리뷰트로 갖는 사실 테이블(fact table)이 있다고 할 때, "수입이 2000만원에서 5000만원 사이인 30, 40, 50대의 고객에 대해, 수입을 4 단계, 나이를 3 단계로 하여 그룹을 나눈 후 각 그룹별로 평균 판매량을 구하라."와 같은 질의를 생각해 볼 수 있다. 이 외에도 센서를 통해 측정된 기온이나 강수량의 분포, 또는 도시의 인구분포 등도 범위 모자이크 질의를 수행할 수 있는 좋은 대상이다.

이 논문에서는 범위 모자이크 질의를 지원하기 위한 새로운 SQL 구문인 mosaic-by 절을 제안한다. mosaic-by 절은 group-by 절과 마찬가지로 레코드를 그룹짓는데 사용하며, 연속적인 도메인에 적용 가능한 구문이다. 또한 이 논문에서는 집계 R-트리(aggregate R-tree)[3-5]를 기반으로 하여 범위 모자이크 질의를 효율적으로 수행하는 알고리즘을 제안한다. 집계 R-트리는 R-트리 중간 노드의 각 엔트리마다 하위 노드의 집계값을 저장하는 형태로, 기존 R-트리의 구조를 그대로 유지하기 때문에 널리 사용되고 있다. 제안하는 알고리즘은 범위 모자이크 질의를 구성하는 여러 개의 범위 집계 질의를 한 번의 트리 순회만으로 수행하며, 집계 R-트리 중간 노드의 집계값을 통해 적은 수의 노드 접근만으로 질의 처리를 할 수 있다.

이 논문의 구성은 다음과 같다. 2절에서는 범위 통계 질의와 집계 R-트리에 대해 살펴본다. 3절에서는 범위 모자이크 질의를 살펴본 후, 집계 R-트리 기반의 질의 처리 알고리즘을 소개한다. 4절에서는 실험 결과를 분석하고 5절에서 결론을 맺는다.

2. 관련 연구

제안하는 범위 모자이크 질의는 OLAP 및 공간 데이타베이스에서 주로 사용되는 범위 통계 질의의 한 종류이며, 이 논문에서는 집계 R-트리를 기반으로 하는 범위 모자이크 수행 기법을 제안한다. 먼저 2.1절에서 범위 통계 질의에 대해 알아본 후, 2.2절에서 집계 R-트리에 대해 살펴보도록 하겠다.

2.1 범위 통계 질의

범위 통계 질의(range statistical query)는 특정 질의 영역 내의 데이타에 대한 통계적인 결과를 반환하는 질의로, 지금까지 연구된 범위 통계 질의로는 범위 집계 질의(range aggregate query)와 범위 상위-k 질의(range top-k query) 등이 있다. 먼저, 범위 집계 질의는 질의 영역 내의 데이타에 대한 합, 평균 등의 집계값을 반환하는 질의이다. 예를 들어 움직이는 자동차의 위치를 저장하는 공간 데이타베이스를 생각해보면, "특정 영역 내에 있는 자동차의 총 대수를 구하라"와 같은 질의가 대표적인 범위 집계 질의의 예가 될 수 있다. 범위 집계 질의를 수행하는 기법은 자료구조에 따라서 두 가지로 분류해 볼 수 있다. 첫 번째는 배열 구조에 기반을 둔 기법으로, 질의 영역 내에 포함되어 있는 전체 셀을 모두 접근하지 않고 범위 집계 질의를 수행하는 것에 초점을 두고 있다. 이러한 배열 기반 기법은 Prefix-sum[6] 기법을 시작으로, 여러 변형 기법들이 소개되었다. 두 번째 기법은 트리 구조에 기반을 둔 기법[3-5]이다. 트리 기법에서는 트리의 중간 노드마다 중간 집계값을 저장하여, 질의 범위 내의 모든 노드를 접근하지 않고도 범위 집계 질의를 수행할 수 있도록 하고 있다.

범위 상위-k 질의는 질의 영역의 데이타 중 상위 k개를 반환하는 질의이다. 예를 들어 각 지점의 위치와 매출액을 저장하는 공간 데이타베이스에서, "특정 영역 내에 있는 지점 중 매출액 기준 상위 10개의 지점을 구하라"와 같은 질의가 범위 상위-k 질의의 예가 될 수 있다. 범위 상위-k 질의 역시 배열 기반 기법과 트리 기반 기법으로 나누어 볼 수 있다. 배열 기반 기법으로는 Loh가 제안한 기법이 있으며[7], 이 기법에서는 데이타 큐브 배열을 여러 개의 영역으로 분할하고, 효율적인 질의 수행을 위해 분할된 각 영역별로 상위 몇 개의 데이타를 유지하는 방법을 사용하였다. 트리 기반 기법으로는 집계 R-트리 기반의 범위 상위-k 질의 처리 기법[8]이 있으며, 이 기법에서는 질의의 효율적인 수행을 위한 우선순위 큐 관리와 단말 노드 구성 기법을 함께 제안하였다.

범위 통계 질의 이외에 OLAP 환경에서 많이 사용되는 분석 질의의 형태로 데이타 큐브(Data Cube) 연산[9]이 있다. 데이타 큐브 연산자는 다차원에 대한 group-by 연산자를 확장한 것으로, 가능한 모든 차원에 대한 group-by 연산을 하나의 연산자로 처리할 수 있다. 이러한 데이타 큐브 연산자를 통해 OLAP에서 주로 사용하는 롤업(roll-up)과 드릴다운(drill-down) 작업을 쉽게 수행할 수 있다. 데이타 큐브 연산은 group-by 연산과 마찬가지로 이산적인 도메인을 대상으로 하고 있다. 이 논문에서는 연속적인 도메인을 대상으로 하는 새로운 그룹 연산자인 mosaic-by와, mosaic-by를 사용한 새로운 범위 통계 질의인 범위 모자이크 질의를 제안한다.

2.2 집계 R-트리

이전 절에서 살펴보았던 트리 기반 기법들은 다차원 데이타를 릴레이션 형태로 저장한 후, R-트리[10]와 같은 트리 기반 인덱스를 통해 데이타를 효율적으로 접근한다. 이러한 트리 기반 구조는 다양한 공간 데이타를 저장할 수 있으며, 비어있는 공간을 저장하지 않기 때문

에 성긴 다차원 데이타를 저장하는데 적합하다. R-트리의 변형인 집계 R-트리(aggregate R-tree)는 자식 노드의 집계값을 부모 노드의 각 엔트리마다 저장하는 구조이다. 이러한 집계 R-트리를 통해 질의 영역내의 모든 데이타를 살펴보지 않고 범위 집계 질의를 효율적으로 수행할 수 있다. 범위 집계 질의의 질의 영역에 완전히 포함되는 노드의 경우, 부모 노드의 엔트리에 저장되어 있는 해당 노드의 집계값을 통해서 질의를 수행하기 때문에, 해당 노드 및 해당 노드의 자식 노드들을 더 이상 탐색할 필요가 없다. 따라서 노드 접근 회수를 줄일 수 있고 질의 영역의 크기에 관계없이 거의 일정한 성능을 보인다는 장점이 있다. MRA R-트리(MRA R-tree)[4], 집계 큐브트리(aggregate cubetree)[3], aR-트리(aR-tree)[5] 등이 이러한 집계 R-트리에 속한다. 이러한 집계 R-트리는 R-트리의 성질을 그대로 따르기 때문에, 범위 집계 질의의 수행 이외에도, 기존 R-트리의 범위 질의, k-NN 질의 등을 그대로 사용할 수 있다는 면에서 널리 사용되고 있다.

그림 1은 중간 노드에 자식 노드의 sum 값을 저장하는 2차원 집계 R-트리의 예이다. 그림의 가장 오른쪽 경로를 따라가 보면서 집계 R-트리의 구조에 대해 알아보기로 하자, 일반 R-트리와 마찬가지로 루트 노드 엔트리인 E_3은 엔트리 자식 노드 N_3을 가리키고 있으며, N_3의 각 엔트리 E_{31}, E_{32}는 각각 자신의 자식 노드 N_{31}과 N_{32}를 가리키고 있다. 집계 R-트리는 이러한 기본적인 R-트리 구조를 기반으로 중간 노드의 각 엔트리마다 자식 노드의 집계값을 저장한다. 예를 들어 N_3의 각 엔트리 E_{31}, E_{32}는 자식 노드 N_{31}과 N_{32}의 sum 값으로 28과 4를 저장하고 있으며, 마찬가지로 E_3는 자식 노드 N_3의 sum 값인 32를 저장하고 있다(E_{31}과 E_{32}의 합).

3. 범위 모자이크 질의

먼저 3.1절에서는 mosaic-by 절과 범위 모자이크 질의를 정의하고, 3.2절에서 집계 R-트리를 기반으로 하는 범위 모자이크 질의의 효율적인 수행 기법에 대해 살펴보도록 하겠다.

3.1 범위 모자이크 질의의 정의

범위 모자이크 질의는 질의 영역을 각 차원별로 여러 개의 구간으로 나누어 만들어진 다차원 격자의 각 영역에 대해 집단 함수를 수행하여, 각 영역별 집계값을 결과로 반환하는 질의이다. 이 때, 만들어진 격자를 모자이크 격자(mosaic grid)라 하며, 모자이크 격자에 의해 구분된 각 영역을 모자이크 셀(mosaic cell)이라고 한다.

이 논문에서는 이러한 범위 모자이크 질의를 SQL문으로 표현하기 위해 mosaic-by 절을 추가하였다. group-by 절과 마찬가지로 mosaic-by 절은 하나 혹은 그 이상의 애트리뷰트에 대해 적용되며, 해당 애트리뷰트의 값에 의해 레코드를 그룹짓게 된다. 하지만 group-by가 애트리뷰트의 값이 같은 것끼리 그룹을 짓는 것에 비해, mosaic-by는 명세한 격자를 통해 같은 셀 내에 포함된 레코드끼리 그룹을 짓는다. mosaic-by 절 역시 group-by 절과 마찬가지로 sum, count 등의 집단함수(aggregate function)와 함께 사용된다.

mosaic-by 절의 구문은 다음과 같다.

MOSAIC BY $d_1(g_1)$, $d_2(g_2)$, ..., $d_n(g_n)$

d_i는 차원 애트리뷰트이고, 각 차원 i는 g_i에 의해서 분할된다. g_i를 표현하는 방법은 두 가지가 있는데, 하나는 단일 값으로 표현하는 방법이다. g_i를 단일 값으로 표현하는 경우, 차원 i는 g_i개의 동일한 크기의 구간으로 분할 된다. 예를 들어 "MOSAIC BY a(3), b(2)"과 같이 표현하면 레코드를 애트리뷰트 a, b에 대해 3×2 형태의 모자이크 격자로 그룹지으라는 의미를 갖는다.

g_i를 표현하는 두 번째 방법은 차원별 격자값의 리스트로 표현하는 것이다. 각 격자값은 ','로 구분지어 나타낸다. 예를 들어 "MOSAIC BY a(10, 15, 50, 100), b(20, 40, 50)"과 같이 표현하면 애트리뷰트 a에 대해

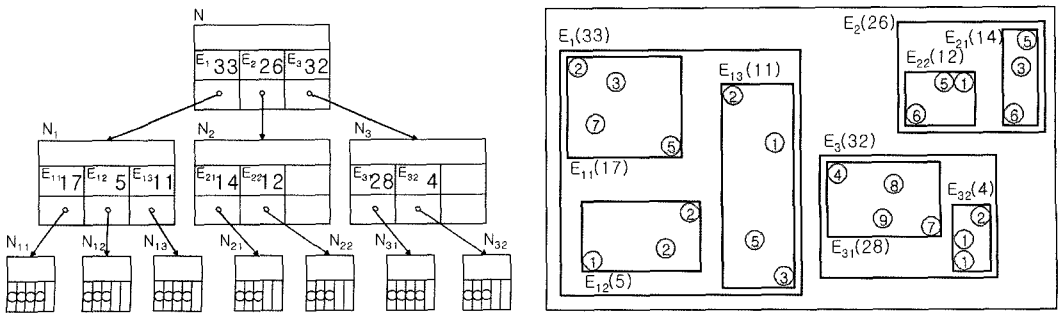


그림 1 집계값으로 sum 값을 저장한 2차원 집계 R-트리

10, 15, 50, 100을 기준으로 격자를 분할하고, b에 대해 20, 40, 50을 기준으로 격자를 분할하게 된다.

이러한 mosaic-by절을 통해 각 레코드는 차원 애트리뷰트 값에 따라 각각 해당되는 모자이크 셀로 그룹지어진다. group-by의 경우에는 차원 애트리뷰트 값이 같은 레코드끼리 그룹지어지지만, mosaic-by의 경우에는 하나의 모자이크 셀로 그룹지어지는 레코드의 차원 애트리뷰트 값들은 서로 다른 값을 가질 수 있다. 이 논문에서는 레코드의 각 차원 애트리뷰트로부터 모자이크 셀의 각 차원별 시작 격자값과 종료 격자값을 구하는 집단함수로 start(d_i), end(d_i)을 제안한다. start(d_i)는 해당 레코드가 포함된 모자이크 셀에 대한 d_i 차원의 시작 격자값을 반환한다. 마찬가지로 end(d_i)는 해당 레코드가 포함된 모자이크 셀에 대한 d_i 차원의 종료 격자값을 반환한다.

이러한 mosaic-by 절과 start, end 함수를 통해 범위 모자이크 질의를 다음과 같이 SQL 문으로 표현할 수 있다.

SELECT start(d_1), end(d_1), start(d_2), end(d_2), ..., start(d_n), end(d_n), aggregate(v)
FROM R
MOSAIC BY $d_1(g_1)$, $d_2(g_2)$, ..., $d_n(g_n)$
WHERE $d_1>=m_1$ and $d_1<=M_1$ and $d_2>=m_2$ and $d_2<=M_2$ and ... and $d_n>=m_n$ and $d_n<=M_n$

테이블 R에 대한 범위 모자이크 질의의 질의 영역은 where 절에 의해 표현된다. mosaic-by 절에 의해 모자이크 격자가 구성되며, 같은 모자이크 셀에 포함된 레코드끼리 그룹지어진다. 마지막으로 select 절에서는 start, end 함수를 통해 모자이크 셀의 각 차원별 시작과 끝 격자값을 반환하고, 집단함수를 통해 합, 평균 등의 집계값을 반환하게 된다.

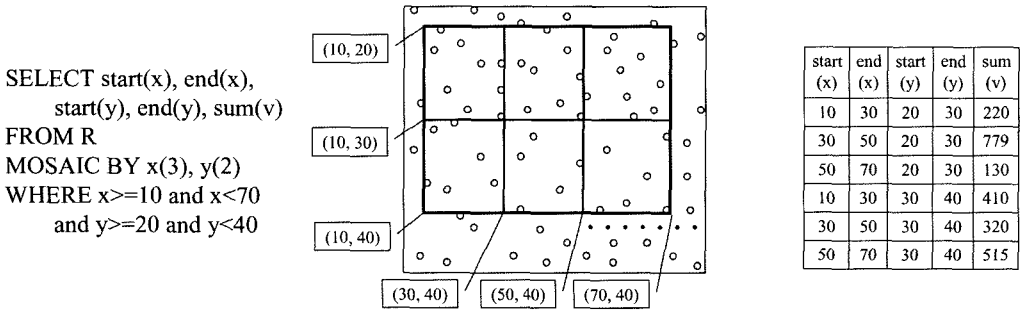
그림 2(a)는 SQL로 표현된 범위 모자이크 질의의 예이다. x, y는 mosaic-by의 대상이 되는 차원 애트리뷰트이고, v는 집단함수 sum의 대상이 되는 값 애트리뷰트이다. 질의 영역은 where절을 통해 명시하게 된다. 명시된 질의 영역은 mosaic-by 절에 의해 그림 2(b)와 같이 3×2 격자로 나뉘고, 격자의 각 셀 내의 레코드끼리 그룹지어 진다. 그런 다음 각 그룹별로 차원 애트리뷰트 x, y에 대한 start, end 함수와 v에 대한 sum 함수가 수행되어 그림 2(c)와 같은 결과가 얻어지게 된다.

3.2 질의 수행

범위 모자이크 질의는 여러 가지 방법으로 수행될 수 있다. R-트리의 범위 질의를 이용하는 기법인 범위 질의 후 집계(Range Query and Aggregation, RQA) 기법, 범위 모자이크 질의를 여러 개의 범위 집계 질의로 분해해서 집계 R-트리를 통해 수행하는 다중 범위 집계 질의(Multiple Range Aggreagate Query, MRAQ) 기법, 집계 R-트리와 큐를 기반으로 범위 모자이크 질의를 효율적으로 수행하는 다중 셀 갱신(Multiple Cell Update, MCU) 기법 등이 그것이다. 이 절에서는 각 기법에 대해 자세히 살펴보기로 하자.

범위 질의 후 집계(RQA) 기법은 R-트리를 통해 해당 범위 질의를 먼저 수행하여 범위 내의 레코드를 결과로 얻어낸 후, 결과 레코드를 차원 애트리뷰트에 대해 정렬하여 각 모자이크 셀마다 집계값을 계산하는 것이다. RQA 기법은 질의 영역을 분할하는 모자이크 셀의 개수와 상관없이 일정한 성능을 낼 수 있으나, 질의 영역 내의 모든 노드를 접근해야 하기 때문에 질의 영역의 크기가 커질수록 성능이 안 좋아지게 되는 단점이 있다.

다중 범위 집계 질의(MRAQ) 기법은 각 모자이크 셀마다 각각의 범위 집계 질의를 수행하는 것이다. MRAQ 기법은 집계 R-트리를 통해 각 범위 집계 질의를 수행하기 때문에[3-5], 질의 영역 내의 모든 노드를 접근 하지 않을 수 있어서 질의 성능을 높일 수 있다. 하지만 모



start (x)	end (x)	start (y)	end (y)	sum (v)
10	30	20	30	220
30	50	20	30	779
50	70	20	30	130
10	30	30	40	410
30	50	30	40	320
50	70	30	40	515

(a) SQL (b) 질의 (c) 결과

그림 2 범위 모자이크 질의의 예

자이크 셀의 개수가 커질 경우 수행해야 할 범위 집계 질의의 수가 늘어나므로 성능이 떨어지게 된다.

다중 셀 갱신(MCU) 기법은 한 번의 트리 순회만으로 질의를 수행하며, 질의 영역 내의 데이타를 모두 접근하지 않고 질의를 수행할 수 있다. 알고리즘은 집계 R-트리와 큐를 이용하여 수행된다. 큐에는 집계 R-트리 노드의 포인터가 유지된다. 알고리즘 1에 MCU 기법의 알고리즘이 나와 있다. 알고리즘의 수행 과정을 살펴보면 다음과 같다. 먼저 루트 노드의 포인터를 큐에 삽입한다(5행). 그 다음, 큐에서 포인터 하나를 뽑는다(7행). 뽑힌 포인터를 p라 하면 p가 가리키는 노드 n을 디스크에서 읽는다(8행). 노드 n의 각 엔트리 e에 대해 다음의 세 가지 경우를 고려해 해당 작업을 수행한다(9행). 먼저 e가 하나의 모자이크 셀 c에 완전히 포함되는 경우에는(10행) e에 저장된 집계값을 해당 모자이크 셀 c의 결과값에 집계한다(11행). 이 경우 e가 가리키는 자손 노드 및 그 후손들은 더 이상 접근할 필요가 없다. 이때, 10행의 e.included(mosaic_grid) 함수는 인자로 받은 mosaic_grid의 각 셀 중에 엔트리 e가 완전히 포함되는 셀이 있는지를 판단하여, 있으면 해당 셀 c를 반환하고 없으면 false를 반환하는 함수이다. 그리고, 11행의 c.aggregate(e.aggregate_value) 함수는 인자로 받은 엔트리 e의 집계값 e.aggregate_value를 셀 c에 집계하는 함수이다. 두 번째로 e가 하나 이상의 모자이크 셀과 겹치는 경우에는(12행) e에 있는 자식 노드의 포인터를 큐에 삽입하여, 나중에 그 자손노드를 접근하도록 한다(13행). 이때, 12행의 e.overlap(mosaic_grid) 함수는 인자로 받은 mosaic_grid의 셀들과 엔트리 e가 겹치는 여부를 판단하는 함수로 겹치면 true를, 아니면 false를 반환한다. 그 이외의 경우에는 아무 작업도 하지 않고 넘어간다. 이러한 과정을 큐가 완전히 빌 때까지 반복하면 된다(16행).

```
1  Algorithm Range_Mosaic
2  Input: 집계 R-트리, 범위 모자이크 질의
3  Output: 각 모자이크 셀에 대한 집계값
4  begin
5     queue.enqueue(p_root_node);
6     repeat
7        p ← queue.dequeue();
8        n ← read_node(p);
9        foreach e in n.entries do
10          if c ← e.included(mosaic_grid) then
11             c.aggregate(e.aggregate_value);
12          else if e.overlap(mosaic_grid) then
13             queue.enqueue(e.child_node_pointer);
14       end if
15    end for
16    until queue.is_empty();
17 end
```

알고리즘 1. 범위 모자이크 질의 알고리즘 (MCU)

3.3 성능 분석

이 절에서는 3.2절에서 제시한 RQA, MRAQ, MCU 기법의 성능에 대한 비용 모델을 통해 성능 분석을 하도록 하겠다. 성능 분석의 편의를 위해 다른 연구[11]에서와 같이 다음과 같은 가정을 하도록 하겠다. 먼저, d 차원 질의 공간 R^d는 $[0..1)^d$이며, 각 데이타 객체는 공간상의 점 데이타이고, 균등 분포를 따른다. 그리고, 같은 레벨에 있는 노드는 모두 같은 크기를 가지며, 각 노드는 초월 정사각형(hyper square)으로 구성된다. 질의 영역은 초월 정사각형으로 구성되며, 모든 차원에 대해 격자의 수는 동일하다. 그림 3은 질의 영역과 노드와의 관계를 나타내는 예이다. 질의 영역은 굵은 점선으로 표시하였으며, 2×2 모자이크 격자에 의해 4개의 정사각형 모자이크 셀로 구성된다. 노드는 실선으로 표시하였으며 역시 정사각형 형태이다. 노드 A는 모자이크 셀 내에 완전히 포함된 경우이며, 노드 B는 모자이크 셀과 부분적으로 겹치는 경우를 나타내고 있다.

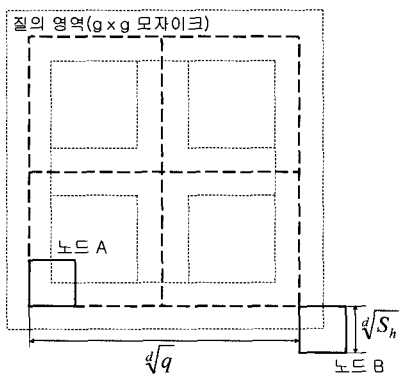


그림 3 질의 영역과 노드와의 관계

먼저 RQA 기법에 필요한 노드 접근 회수를 알아보도록 하겠다. RQA 기법은 전체 질의 영역에 대해 범위 질의를 수행해야 하므로, 전체 질의영역과 겹치는 노드의 수를 구하면 된다. 전체 레코드의 수를 N, 노드의 팬아웃을 f라고 하면, 레벨 h에서의 노드 개수 N_h는 다음과 같은 식으로 나타낼 수 있다.

$$N_h = \lceil \frac{N}{f^h} \rceil$$

노드간의 겹침이 없고, 동일 레벨의 모든 노드가 전체 영역을 커버한다고 가정하면, 레벨 h에서의 노드 크기 S_h는 다음과 같다.

$$S_h = \frac{1}{N_h}$$

질의 영역의 크기를 q, 레벨 h의 노드 중에 질의 영역과 겹치는 노드의 개수를 $O_h(q)$라고 하자. 민코프스키 합(Minkowski Sum) 원리에 의해 레벨 h의 한 노드가 질의 영역과 겹치는 확률은 $(\sqrt[d]{q}+\sqrt[d]{S_h})^d$이다 [11]. 따라서, $O_h(q)$는 다음과 같이 구할 수 있다.

$$O_h(q) = (\sqrt[d]{q}+\sqrt[d]{S_h})^d N_h$$

따라서 RQA 기법에 필요한 노드 접근 회수 RQA는 다음과 같이 구할 수 있다. 이 때 H는 트리의 높이이며, $H = \lceil \log_f N \rceil$ 으로 나타낼 수 있다.

$$RQA = \sum_{h=1}^{H} O_h(q)$$

$$= \sum_{h=1}^{\lceil \log_f N \rceil} (q^{\frac{1}{d}} + \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

이번에는 MRAQ 기법에 필요한 노드 접근 회수를 구해보자. 이를 위해 집계 R-트리를 사용하여 전체 질의 영역에 대한 범위 집계 질의를 수행하는데 필요한 노드 접근 회수 RAQ를 구해보도록 하자. 레벨 h의 노드 중에 크기 q인 질의 영역에 완전히 포함되는 노드의 개수를 $I_h(q)$라고 하면, 마찬가지 방법으로 레벨 h의 한 노드가 질의 영역에 완전히 포함되는 확률은 $(\sqrt[d]{q}-\sqrt[d]{S_h})^d$이다. 따라서, $I_h(q)$는 다음과 같이 구할 수 있다.

$$I_h(q) = (\sqrt[d]{q}-\sqrt[d]{S_h})^d N_h$$

범위 집계 질의에서는 질의 영역에 완전히 포함되는 노드의 하위 노드는 접근할 필요가 없으므로, RAQ는 다음과 같은 식으로 나타낼 수 있다.

$$RAQ = \sum_{h=1}^{H} (O_h(q) - fI_{h+1}(q)) \qquad (I_{H+1}(q) = 0)$$

$$= \sum_{h=1}^{\lceil \log_f N \rceil} (q^{\frac{1}{d}} + \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

$$- f(q^{\frac{1}{d}} - \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

MRAQ 기법은 전체 질의 영역을 각 셀에 해당하는 여러 개의 작은 질의 영역으로 나눈 후 각 영역에 대해 범위 집계 질의를 수행하게 된다. 차원별 격자의 개수를 g라고 하면 각 셀의 크기는 $\frac{q}{g^d}$로 나타낼 수 있다. 따라서 MRAQ 기법에 필요한 노드 접근 회수 $MRAQ$는 다음과 같이 구할 수 있다.

$$MRAQ = g^d \sum_{h=1}^{H} (O_h(\frac{q}{g^d}) - fI_{h+1}(\frac{q}{g^d})) \qquad (I_{H+1}(q) = 0)$$

$$= \sum_{h=1}^{\lceil \log_f N \rceil} (\frac{q^{\frac{1}{d}}}{g} + \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

$$- f(\frac{q^{\frac{1}{d}}}{g} - \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

마지막으로 MCU 기법에 필요한 노드 접근 회수 MCU를 알아보도록 하겠다. 레벨 h의 노드 중에 크기가 q이고, 차원별 격자수가 g인 질의를 구성하는 모자이크 셀 중 하나에 완전히 포함되는 노드의 개수를 $C_h(q)$라 하면, 레벨 h의 한 노드가 모자이크 셀 내에 완전히 포함될 확률은 $g^d(\frac{\sqrt[d]{q}}{g} - \sqrt[d]{S_h})^d$로 나타낼 수 있으므로, $C_h(q)$는 다음과 같이 구할 수 있다.

$$C_h(q) = g^d(\frac{\sqrt[d]{q}}{g} - \sqrt[d]{S_h})^d N_h$$

MCU 기법에서는 질의 영역을 구성하는 모자이크 셀 내에 완전히 포함되는 노드의 경우 해당 노드의 하위 노드는 접근할 필요가 없으므로 MCU는 다음과 같이 나타낼 수 있다.

$$MCU = \sum_{h=1}^{H} (O_h(q) - fC_{h+1}(q)) \qquad (C_{H+1}(q) = 0)$$

$$= \sum_{h=1}^{\lceil \log_f N \rceil} (q^{\frac{1}{d}} + \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

$$- fg^d(\frac{q^{\frac{1}{d}}}{g} - \lceil \frac{N}{f^h} \rceil^{-\frac{1}{d}})^d \lceil \frac{N}{f^h} \rceil$$

그림 4는 $f = 170$, $d = 2$, $N = 1{,}000{,}000$인 집계 R-트리에서 $g = 20$인 범위 모자이크 질의를 수행할 때 필요한 노드 접근 회수를 추정해 본 것이다. 비용 함수를 통해 얻은 추정치와 실험을 통해 얻은 실측값이 거의 비슷한 것을 확인할 수 있다. RQA 기법의 경우 질의 영역의 크기가 커짐에 따라 노드 접근 회수가 급격히 증가하는 반면, MRAQ 기법과, MCU 기법은 거의 일정

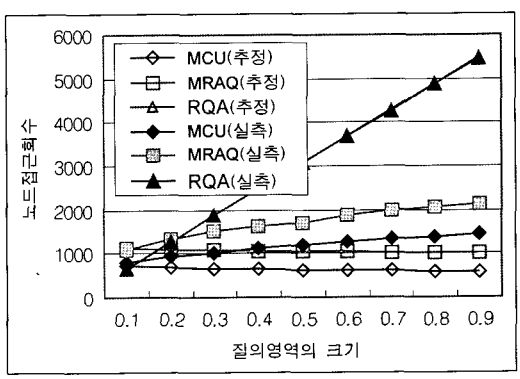


그림 4 비용 모델을 통한 성능 분석

한 노드 접근 회수를 보인다. 또한, MCU기법이 MRAQ 기법에 비해 노드 접근 회수가 적은 것을 알 수 있다. MRAQ 기법과 MCU 기법의 경우 질의 영역이 커짐에 따라 노드 접근 회수가 조금 작아지는 경향을 보이는데, 실제로는 노드간의 겹침이 발생하기 때문에 질의 영역이 커짐에 따라 노드 접근 회수가 증가하게 된다.

4. 실험 및 분석

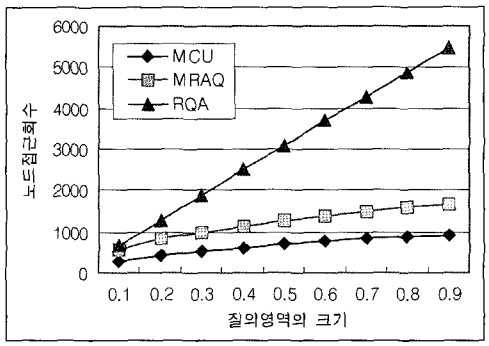
4.1 실험 환경

3.2절에서 제시한 RQA, MRAQ, MCU 기법의 성능을 실험을 통해 비교 평가하였다. 다양한 종류의 분석을 위해, 2차원부터 4차원까지의 균등(uniform) 분포를 따르는 데이타셋 U2, U3, U4를 사용하였으며, 각 데이타셋은 100만 개의 데이타로 구성하였다. 각 데이타셋에 대해 질의 영역의 크기, 모자이크의 개수, 데이타의 차원 및 레코드 수 등을 변화 시켜가면서 실험을 하였다. 그 외에도 실제 데이타셋 및 실제 데이타 분포를 반영한 데이타셋 등에 대한 실험을 위해 TPC-H [12]에서 사용하는 데이타셋 H와 Tiger/Line [13]의 실제 데이타셋 TL을 사용하였다. H는 TPC-H의 스키마 중 orders 테이블의 레코드 150만개를 생성하여 사용하였으며, TL은 Tiger/Line 데이타 중 2002년의 미국 뉴욕 지역의 데이타로, 레코드 수는 약 140만 개인 2차원 데이타이다.

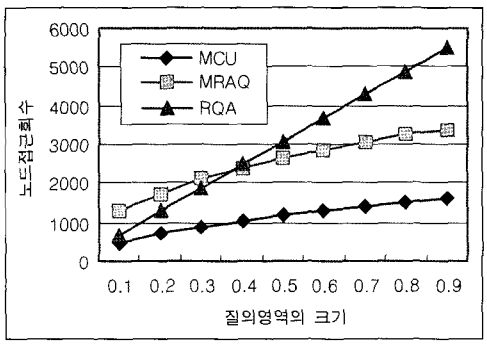
각 실험은 주메모리 512MB, 하드 40GB인 펜티엄3 1GHz 시스템에서 수행하였다. R-트리와 집계 R-트리의 페이지 크기는 4KB로 하였다.

4.2 실험 결과

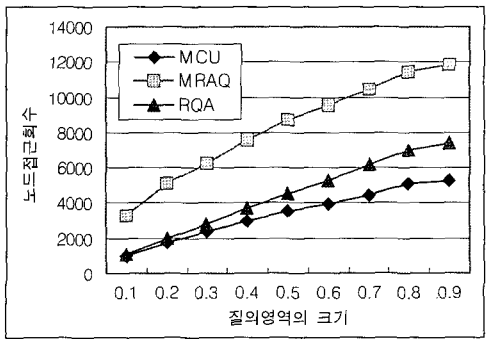
RQA, MRAQ, MCU 기법의 성능을 비교하기 위해, 먼저 U2, U3, U4 데이타셋에 대해 질의 영역의 크기를 변화시켜가면서 범위 모자이크 질의를 수행하였다. 그림 5는 각 데이타셋에 대한 실험 결과이다. 질의 영역의 크기는 전체 초월 평면(hyper-plane)의 크기에 대한 질의 영역 초월 평면 크기의 비율로 나타내었다. (a)는 U2 데이타셋에 대해 5×5 모자이크를, (b)는 10×10 모자이크를 사용한 결과이며, (c)는 U3 데이타터셋에 대해 5×5×5 모자이크를, (d)는 U4 데이타셋에 대해 5×5×5×5 모자이크를 사용한 결과를 보여주고 있다. 먼저 2차원 데이타에 대한 실험인 (a)(b)에 대해 살펴보면, RQA 기법의 경우 질의 처리를 위해 질의 영역내의 모든 노드를 접근해야 하므로, 질의 영역의 크기가 커짐에 따라 노드 접근 회수가 급속히 증가하는 결과를 보인다. 이에



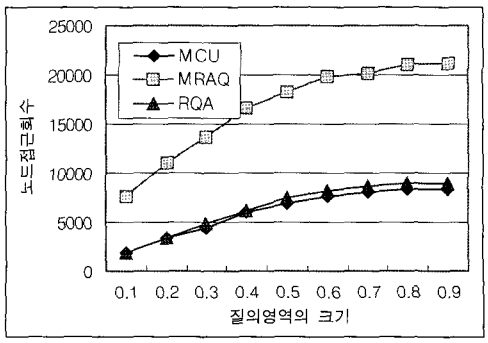
(a) 2차원, 5×5 모자이크



(b) 2차원, 10×10 모자이크



(c) 3차원, 5×5×5 모자이크



(d) 4차원, 5×5×5×5 모자이크

그림 5 질의 영역 크기의 변화에 따른 노드 접근 회수

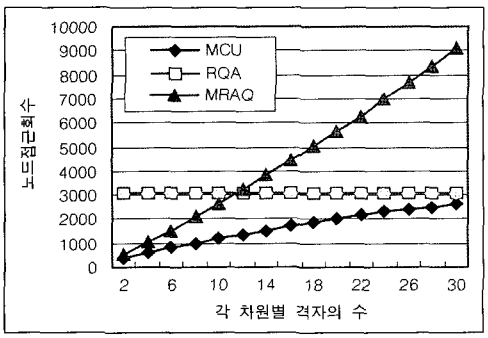
비해 MCU 기법은 질의 영역의 크기가 커지더라도 노드 접근 회수가 크게 증가하지 않는 것을 볼 수 있다. 이는 집계 R-트리의 집계값을 이용하여 질의 영역 내의 일부 노드만을 접근하고 질의를 수행할 수 있기 때문이다. 또한 MCU 기법의 성능이 항상 MRAQ 기법보다 좋은 것을 알 수 있다. MRAQ 기법은 모자이크 셀 개수 만큼 범위 집계 질의를 수행해야 하기 때문에, 10×10 모자이크의 경우 RQA 기법보다도 성능이 떨어지는 구간이 존재한다. 2, 3차원 데이타에 대한 실험인 (c), (d)의 경우, 차원 증가에 따라 모자이크 셀의 개수가 증가하여, MRAQ 기법의 노드 접근 회수가 크게 증가한 것을 볼 수 있다. MCU 기법이 가장 좋은 성능을 보이나 차원이 증가할수록 성능의 차이가 크게 두드러지지는 않는다.

*모자이크 셀의 개수*는 *범위 모자이크 질의*에 있어서 매우 중요한 요소이다. U2 데이타셋에 대해 모자이크 셀의 개수를 변화시켜 가면서 범위 모자이크 질의를 수행하였다. 질의 영역의 크기는 50%로 고정시키고, 각 차원별 격자의 수를 변화시켜 가면서 노드 접근 회수를 측정하였다. 그림 6은 모자이크 셀의 개수 변화에 따른 세 기법의 성능을 보여주고 있다. 그래프에서 차원별 격자수는 모자이크 셀의 개수를 나타낸다. 예를 들어 차원별 격자수가 5이면, 5×5 모자이크가 사용된 것을 의미한다. 각 차원별로 격자의 수가 다를 수 있지만 이 실험에서는 편의상 모든 차원의 격자수를 같게 구성하였다. *질의 영역의 크기가 일정할 경우, 차원별 격자수가 클수록* 모자이크 셀의 개수는 커지고, 각 모자이크 셀의 크기는 작아진다. 반대로, 차원별 격자수가 작아질수록 모자이크 셀의 개수는 작아지고, 각 모자이크 셀의 크기는 커진다.

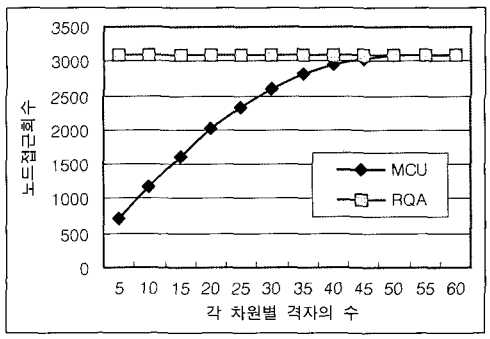
전체 질의 영역의 크기가 고정되어 있는 경우, RQA 기법은 모자이크의 셀의 개수와 관계없이 일정한 수의 노드를 접근하게 된다. 이는 RQA 기법에서는 전체 질의 영역에 대해 범위 질의를 수행해야 하므로 모자이크 셀의 개수와 관계없이 질의 영역 내의 모든 노드를 접근해야 하기 때문이다. 반면에 MRAQ 기법과 MCU 기법의 경우 모자이크 셀의 개수에 의해 영향을 받는다. MRAQ 기법의 경우 모자이크 셀의 개수만큼 범위 집계 질의를 수행해야 하기 때문에, 모자이크 셀의 개수가 높아질수록 질의 성능이 급격히 떨어진다. 또한 하나 이상의 모자이크 셀과 겹치는 노드의 수가 증가하기 때문에 하나의 노드를 두 번 이상 접근하는 일이 많아진다. 그에 비해 MCU 기법은 모자이크 셀의 개수가 높아지더라도 노드 접근 회수가 급격하게 증가하지는 않는다. MCU 기법에 있어서 최악의 경우는 질의 영역 내의 모든 노드 중에서 모자이크 셀에 완전히 포함되는 노드가 하나도 없는 경우이다. 이 경우에는 질의 영역 내의 모든 노드를 접근해야 하지만, 하나의 노드를 두 번 이상 방문하지 않으므로, 최소한 RQA기법보다는 성능이 나빠지지 않는다.

MCU 기법의 확장성을 살펴보기 위해 레코드의 개수와 차원을 변화시키면서 실험을 수행하였다. 먼저 U2 데이타셋의 레코드의 수를 10만개부터 1000만개까지 변화시키면서 노드 접근 회수를 측정하였다. 질의 영역은 50%로 고정시켰으며, 10×10 모자이크를 사용하였다. 그림 7(a)는 레코드의 개수별 노드 접근 회수를 측정한 결과이다. RQA 기법은 레코드 수가 작은 경우에는 좋은 성능을 보이나, 레코드 수가 증가할수록 성능이 급격히 나빠지는 것을 알 수 있다. MCU 기법은 레코드 수가 커짐에 따라 완만한 노드 접근 회수의 증가를 보이며, 모든 경우에 있어서 RQA 기법과 MRAQ 기법보다 좋은 성능을 보이는 것을 알 수 있다.

MCU 기법의 확장성을 살펴보기 위한 두 번째 방법으로 차원을 변화시켜가면서 노드 접근 회수를 측정하였다. 2~4 차원의 데이타셋 U2, U3, U4를 사용하였으며, 질의 영역의 크기를 50%로 고정하고, 모든 차원에

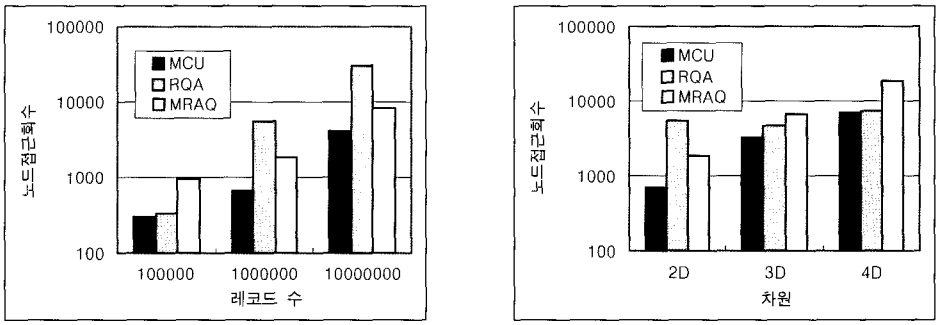


(a) MCU, RQA, MRAQ



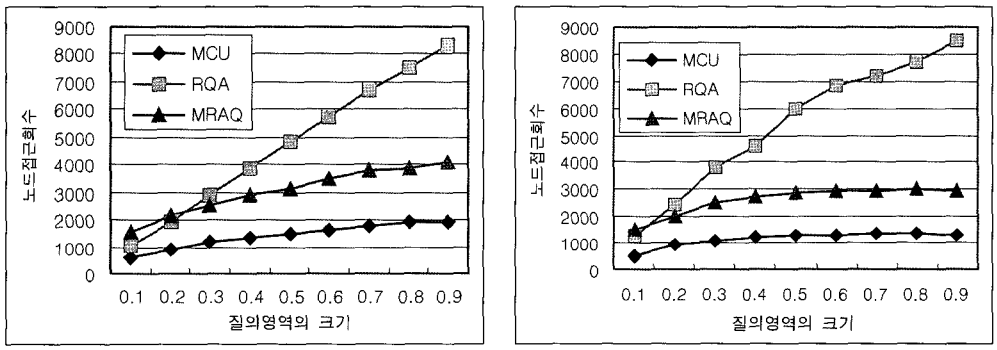
(b) MCU, RQA

그림 6 모자이크 셀의 개수 변화에 따른 노드 접근 회수



(a) 레코드 수 (b) 차원

그림 7 레코드 수와 차원의 변화에 따른 노드 접근 회수



(a) TPC-H 데이타셋(H) (b) Tiger Line 데이타셋(TL)

그림 8 다양한 데이타셋에 대한 범위 모자이크 질의 수행

대해 거의 같은 개수의 모자이크 셀을 유지하였다. 즉 U2에 대해서는 9×9 모자이크 격자를, U3에 대해서는 4×4×5 모자이크 격자를 사용하였고, U4에 대해서는 3×3×3×3 모자이크 격자를 사용하였다. 그림 7(b)는 각 차원별 노드 접근 회수를 측정한 결과이다. 결과에서 알 수 있듯이, 일반적인 R-트리가 사용되는 저차원의 경우 MCU는 만족할만한 성능을 보인다.

범위 모자이크 질의를 다양한 데이타셋(H, TL)을 통해 수행해 보았다. 그림 8은 각 데이타셋에 대해 질의영역의 크기를 변화시켜 가면서 10×10 모자이크 질의를 수행한 결과이다. 실제 데이타 분포를 반영하는 데이타셋(H) 및 실제 데이타셋(TL)에서도 생성된 데이타와 비슷한 결과를 보이는 것을 알 수 있다.

범위 모자이크 질의는 고정 그리드(fixed grid)기반의 자료구조를 통해서도 수행가능하다. 그림 9는 고정 그리드에 계층적인 집계값을 부가적으로 저장하여 MCU 알고리즘을 수행한 결과이다. 100만개의 U2 데이타셋을 사용하였으며, 10×10 모자이크 질의에 대해 질의 영역의 크기를 변화해 가면서 노드 접근 회수를 측정하였다. 고정 그리드를 사용하는 경우 생성 전에 그리드의 크기를 결정하여야 하므로, 100×100, 50×50, 63×63 세 종류

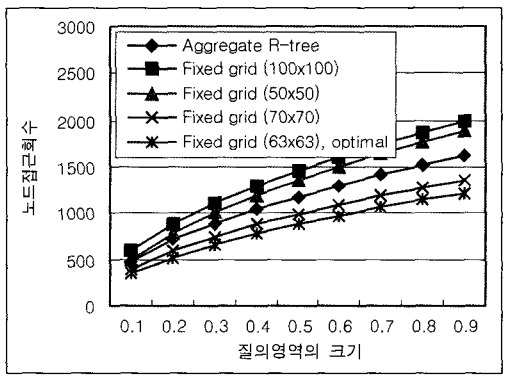


그림 9 고정 그리드와의 비교

의 그리드를 사용하였다. 페이지 크기가 4 kbyte이고, 레코드 하나가 16 byte로 구성되므로, 한 페이지에는 256개의 레코드를 저장할 수 있다. 100×100 그리드의 경우 평균적으로 그리드 버킷 하나당 100개의 레코드를 저장하면 되기 때문에, 전체적으로 오버플로우가 거의 발생하지 않는 반면, 버킷을 구성하는 페이지 수가 증가하여 전체적으로 집계 R-트리보다 성능이 떨어지게 된다. 50×50 그리드의 경우 버킷의 수는 적지만 버킷 하

나당 400개의 레코드를 저장해야 하므로, 대부분의 버킷에서 오버플로우가 발생하여 전체적인 페이지수는 100×100 그리드와 거의 비슷하다. 균등 분포 데이타이므로 100만개의 데이타에 대한 최적의 그리드를 구해보면 63×63 그리드이며, 이 경우 집계 R-트리보다 좋은 성능을 보이는 것을 알 수 있다. 70×70 그리드의 경우 최적인 그리드와 거의 비슷한 성능을 보인다. 집계 R-트리의 경우에는 데이타의 개수와 분포에 상관없이 트리를 균형적으로 유지하며, 데이타 페이지 구성을 효율적으로 하기 때문에 최적화되지 않은 그리드보다 좋은 성능을 보인다. 따라서 데이타의 분포와 개수를 미리 예상할 수 있는 응용의 경우 최적화된 고정 그리드를 사용하는 것이 적당하며, 데이타의 분포와 개수에 대해 예측할 수 없거나, 동적으로 변하는 환경에서는 집계 R-트리를 사용하는 것이 적당하다.

5. 결 론

이 논문에서는 새로운 형태의 범위 통계 질의인 범위 모자이크 질의를 제안하였다. 범위 모자이크 질의는 주어진 질의 영역에 대해 다차원 모자이크 격자를 적용하여 여러 개의 모자이크 셀로 그룹짓고, 각 모자이크 셀에 대해 집단 함수를 수행하는 질의이다. 범위 모자이크 질의는 제안한 mosaic-by 절을 통해 SQL문으로 표현 가능하다.

또한 이 논문에서는 집계 R-트리를 기반으로 하는 효율적인 범위 모자이크 질의 알고리즘을 제안하였다. 집계 R-트리는 다차원 데이타의 색인 기법으로 널리 사용되고 있으며, 범위 질의, 범위 집계 질의 등 다차원 데이타에 대한 다양한 질의를 처리할 수 있는 구조이다. 알고리즘은 각 모자이크 셀의 집계값을 한번의 트리 순회만으로 계산하며, 집계 R-트리의 집계값을 이용하여 질의 영역 내의 모든 노드를 접근하지 않고도 질의를 수행할 수 있도록 하였다. 따라서 작은 수의 노드 접근만으로도 질의를 수행할 수 있다. 실험 결과에서 알 수 있듯이, 제안된 알고리즘은 다양한 질의 영역의 크기, 모자이크의 셀의 개수, 레코드의 수, 차원에 대해 만족할 만한 결과를 보인다.

10차원이 넘는 고차원 데이타의 경우 R-트리와 같은 트리구조를 사용하면 오히려 순차 검색보다도 성능이 나빠지게 된다[11]. 따라서 고차원에서는 이 논문에서 제시한 집계 R-트리 기반의 범위 모자이크 질의를 그대로 적용할 수 없다. 또한, 데이타의 차원이 증가함에 따라 모자이크 셀의 개수는 기하급수적으로 증가하게 되는데, 예를 들어 10차원의 경우 각 차원별 격자의 수가 2개만 되어도 1024개의 셀이 생기게 된다. 따라서 고차원의 경우에는 범위 모자이크 질의의 결과 레코드 중 상위 k 개의 결과를 구하는 질의인 범위 모자이크 상위-k 질의가 의미를 갖는다. 현재 추후 연구로, 다차원 배열 기반의 범위 모자이크 질의와 범위 모자이크 상위-k 질의에 대한 연구를 수행 중에 있다.

참 고 문 헌

[1] Ralf Hartmut Guting, "An Introduction to Spatial Database Systems," VLDB Journal Vol.3, No.4, pp.357-399, 1994.

[2] Surajit Chaudhuri, Umeshwar Dayal, "An Overview of Data Warehousing and OLAP Technology," SIGMOD Record, Vol.26, No.1, pp.65-74, 1997.

[3] Seokjin Hong, Byoungho Song, and Sukho Lee, "Efficient execution of range aggregate queries in data warehouse environments," Conceptual Modeling-ER 2001: 20th International Conference on Conceptual Modeling, Yokohama, Japan, pp.299-310. Springer, 2001.

[4] Iosif Lazaridis and Sharad Mehrotra, "Progressive approximate aggregate queries with a multi-resolution tree structure," Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, California, May 21-24, 2001, pp.401-412. ACM Press, 2001.

[5] Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao, "Efficient OLAP operations in spatial data warehouses," In Advances in Spatial and Temporal Databases : 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings, pp.443-459. Springer, 2001.

[6] Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant, "Range queries in OLAP data cubes," Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA, pp.73-88. ACM Press, 1997.

[7] Zheng Xuan Loh, Tok Wang Ling, Chuan-Heng Ang, and Sin Yeung Lee, "Adaptive method for range top-k queries in OLAP data cubes," DEXA'02, pp.648-657. Springer, 2002.

[8] Seokjin Hong, Bongki Moon, and Sukho Lee, "Processing range top-k queries in a sparse data cube," Proceedings of the International Conference on Information and Knowledge Engineering. CSREA Press, 2004.

[9] J. Gray, A. Bosworth, A. Layman, and H. Piramish, "Data Cube: A Relational Aggregation Operator Generalizing, Group-By, Crosstab, and Sub-Totals," Int. Conference on Data Engineering pp.152-159, 1996.

[10] Antonin Guttman, "R-trees: A dynamic index structure for spatial searching," SIGMOD'84, Proceedings ACM SIGMOD International Conference on

Management of Data, June 18-21, 1984, Boston, Massachusetts, pp.47-57. ACM Press, 1984.
[11] S. Berchtold, C. Böhm, and H.P. Kriegel, "The pyramid-technique: Towards breaking the curse of dimensionality," Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA, ed. L.M. Haas and A. Tiwary, pp.142.153, ACM Press, 1998.
[12] TPC-H. http://www.tpc.org/tpch/, 2004.
[13] TIGER/Line. http://www.census.gov/geo/www/tiger/, 2002.

홍 석 진
1998년 서울대학교 컴퓨터공학과 졸업(공학사). 2000년 서울대학교 대학원 컴퓨터공학과 졸업(공학석사). 2000년~현재 서울대학교 대학원 전기컴퓨터공학부 박사과정. 관심분야는 데이타웨어하우스, OLAP, 시공간 데이타베이스, XML

배 진 욱
한국정보과학회 논문지 : 데이타베이스
제 32 권 제 3 호 참조

이 석 호
한국정보과학회 논문지 : 데이타베이스
제 32 권 제 3 호 참조