공간 순서화 곡선을 이용한 다차원 영역 질의 처리†

A Multi-dimensional Range Query Processing using Space Filling Curves

백 현* / Hyun Back
윤지희** / Jee-Hee Yoon
원정임*** / Jung-Im Won

요 약

다차원 공간 객체를 위한 영역 질의는 다차원 공간 상에서 질의 영역과 교차 또는 포함되는 객체들을 검색하는 가장 기본적인 공간 연산이다. 영역 질의 처리를 위한 인덱스 기법으로서 공간 순서화 곡선을 이용하여 다차원 공간 객체의 MBR 정보를 1차원 값으로 변환하여 저장하는 DOT(DOuble Transformation) 인덱스 기법이 알려져 있다. 이 기법은 데이터베이스 시스템의 주색인 기법을 그대로 적용할 수 있는 장점을 갖으나, 중간 공간에 설정된 다차원 질의 영역을 최종 공간상의 1차원 값의 집합으로 변환하는 공간 변환 연산에 대한 오버헤드가 매우 크다는 문제점이 있으며, 원 공간을 2차원 이상으로 확장하여 적용할 수 있는 구체적인 영역 질의 방법이 연구된 바 없다. 본 논문에서는 다차원 공간 질의 영역 상의 공간 순서화 곡선의 규칙성을 분석함으로써 공간 변환 연산의 횟수를 대폭 감소시킨 효율적인 다차원 공간 영역 질의 처리 기법을 제안한다. 제안된 기법에서는 공간 변환 연산의 비용을 감소시키기 위하여 질의 영역을 공간 순서화 곡선이 연속 운행되는 최대 크기의 쿼터로 분할하는 쿼터 분할 기법을 사용한다. 제안된 기법에 의한 다차원 영역 질의 처리 과정을 시각적으로 확인할 수 있는 시뮬레이터를 구현하였으며, 이를 이용한 성능 평가 결과를 보였다.

Abstract

Range query is one of the most important operations for spatial objects, it retrieves all spatial objects that overlap a given query region in multi-dimensional space. The DOT(DOuble Transformation) is known as an efficient indexing methods, it transforms the MBR of a spatial object into a single numeric value using a space filling curve, and stores the value in a B^+-tree. The DOT index is possible to be employed as a primary index for spatial objects. However, the range query processing based on the DOT index requires much overhead for spatial transformations to get the query region in the final space. Also, the detailed range query processing method for 2-dimensional spatial objects has not been studied yet. In this paper, we propose an efficient multi-dimensional range query processing technique based on the DOT index. The proposed technique exploits the regularities in the moving patterns of space filling curves to divide a query region into a set of maximal sub-regions within which

† 이 논문은 2006년도 한림대학교 교비연구비(HRF-2006-030)에 의하여 연구되었음.
■ 논문접수 : 2006.8.8 ■ 심사완료 : 2006.9.11

* 한림대학교 컴퓨터공학과 박사과정(backhyun@sysgate.co.kr)
** 교신저자 한림대학교 정보통신공학부 교수(jhyoon@hallym.ac.kr)
*** 한양대학교 정보통신학부 연구교수(jiwon@hanyang.ac.kr)

space filling curves traverse without interruption. Such division reduces the number of spatial transformations required to perform the range query and thus improves the performance of range query processing. A visual simulator is developed to show the evaluation method and the performance of our technique.

주요어 : 영역 질의, 공간 색인기법, 공간 순서화 곡선, DOT 색인
Keyword : range query, spatial index, space filling curve, DOT index

1. 서 론

다차원 공간 데이터를 이용한 응용 프로그램이나 서비스의 활용이 증가하고 있다. 대표적으로 지리정보시스템(Geographic Information Systems: GIS)이나 위치 기반 서비스(Location Based Service: LBS), 유비쿼터스 환경으로 대변되는 센서 네트워크의 실시간 정보 처리 등을 들 수 있다. 이러한 시스템들은 공간 및 비 공간 데이터들을 동일 시스템 내에서 효율적으로 관리하는 일종의 데이터베이스 시스템으로 볼 수 있다. 전통적인 데이터베이스 시스템을 기반으로 이러한 시스템들을 구현하는 경우 데이터베이스 시스템이 가지고 있는 대용량의 저장, 검색, 질의 최적화, 동시성 제어 및 고장으로부터의 회복 등 다양한 기능을 그대로 계승 받을 수 있으므로 안정성과 확장성, 그리고 타 시스템과의 통합에 있어 유리한 장점을 가지게 된다. 그러나 전통적인 데이터베이스 시스템에서는 제공되지 않는 점(point), 선(line), 면(polygon)과 같은 복합 구조를 가지는 공간 객체를 표현하는 방법과 다차원 공간 객체를 효율적으로 액세스할 수 있는 색인 방법 및 공간 연산자를 필수적으로 제공하여야 한다.

공간 데이터베이스 시스템에서 사용되는 중요한 공간 연산자로는 점 질의(point query), 영역 질의(range query), 공간 조인(spatial join) 등을 들 수 있으며, 이 중 영역 질의는 질의 영역과 교차(intersection) 또는 포함(containment)되는 공간 객체의 집합을 검색하는 가장 기본적인 공간 연산으로 볼 수 있다. 효율적인 공간 연산을 제공하기 위해서는 효율적인 색인 기법의 선택이 중요하며, 색인 기법으로는 R-tree[1, 2, 3], Grid File[4, 5] 등 다차원 색인을 이용한 방법이 알려져 있다.

본 논문에서는 DOT(DOuble Transformation) 공간 색인[6, 7]을 이용한 효율적인 다차원 공간 영역 질의 처리 기법을 제시한다. DOT 공간 색인 기법은 원 공간(initial space)에 존재하는 공간 객체의 최소 경계 사각형(Minimum Bounding Rectangle: MBR) 정보를 공간 순서화 곡선(space filling curve)을 사용하여 하나의 1차원 값으로 변환한 후 그 값을 검색 키(search key)로 갖는 B^+-tree 색인 구조를 구성하는 방법으로서, 이를 이용하면 전통적인 데이터베이스의 주 색인을 적용할 수 있다는 중요한 특징을 가진다.

그러나 원 공간을 2차원 이상으로 확장하여 적용할 수 있는 구체적인 방법이 연구된 바 없으며, 특히 영역 질의 처리를 위하여 중간 공간에 설정된 다차원 질의 영역을 최종 공간상의 1차원 값의 집합으로 변환하는 공간 변환 연산에 대한 오버헤드가 매우 크다는 문제점이 있다.

이러한 문제점 해결을 위하여 우리는 선행 연구[8]를 통하여 1차원 공간 객체의 MBR 정보를 하나의 1차원 값으로 변환하기 위해 사용되는 공간 순서화 곡선의 규칙성

을 분석함으로써 공간 변환 연산의 횟수를 대폭 감소시키는 새로운 기법을 제안한 바 있다. 제안된 기법에서는 반복적으로 수행되는 공간 변환 연산의 횟수를 줄이기 위하여 질의 영역을 공간 순서화 곡선이 연속 운행하는 가능한 최대 크기의 면적으로 분할하는 쿼터 분할 기법을 사용하였다.

본 논문에서는 2차원 공간 객체를 위한 DOT 색인 기반의 효율적인 영역 질의 알고리즘을 제안한다. 우선, 4차원 중간 공간 해석 모델을 제시하여 기존의 DOT 공간 색인 기법이 2차원 이상으로 확장될 수 있음을 보인다. 또한 2차원으로 확장된 영역 질의 알고리즘을 제시하여 쿼터 분할 방법을 4차원 이상의 중간 공간에 적용하는 방법과 차원 확장에 따른 오버헤드를 개선할 수 있는 방법을 제시한다. 또한 연구 평가를 위하여 영역 질의 시뮬레이터를 구현하여 1차원 및 2차원 공간 객체에 대하여 DOT 공간 색인 기법을 이용한 영역 질의 과정을 가시적으로 보이며, 이를 이용한 성능 평가 결과를 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 공간 색인 기법과 관련된 기존의 연구들을 살펴보고, DOT 공간 색인 기법을 간단히 설명한다. 3장에서는 1차원 원 공간 객체에 대하여 DOT 공간 색인 기법을 적용한 영역 질의 방법 및 공간 변환 연산 횟수를 감소시키기 위한 쿼터 분할 방법에 대하여 설명한다. 4장에서는 DOT 공간 색인 기법을 2차원으로 확장하기 위하여 4차원 중간 공간 해석 모델을 제시하고, 2차원 객체의 영역 질의 방법과 알고리즘을 제시한다. 5장에서는 제안된 방식의 유용성과 효율성을 보이기 위하여 시뮬레이터를 이용한 성능 분석 결과를 보인다. 6장에서는 본 연구의 결론을 내린다.

2. 관련 연구

점 질의, 영역 질의, 공간 조인 등의 공간 연산을 위한 다수의 공간 액세스 기법이 제안되어 있다[9, 10]. 효율적인 공간 연산을 위하여 연산 과정은 일반적으로 여과와 정제의 두 단계로 나뉘어 진다[11]. 여과 단계(filter step)는 최종 결과로 남을 공간 객체 후보를 걸러내기 위하여 공간 객체를 근사적으로 표현한 MBR을 사용한 단계이며, 정제(refinement step)단계는 여과 단계에서 얻어진 후보 객체 쌍에 대하여 정밀한 기하 연산을 적용, 최종 연산 결과를 산출해내는 단계이다[3, 11].

공간 객체의 MBR을 이용한 공간 색인 기법은 크게 3가지로 분류 할 수 있다. 첫 번째 방법은 k차원의 공간 객체를 2k차원의 점으로 변환하여, 저장하는 방식으로 Grid-file[4], KDB-tree[12], LSD-tree[13] 등이 이에 속한다. 두 번째 방법은 공간 순서화 곡선(space filling curve)을 사용하여 k차원의 공간 객체를 일련의 1차원 점으로 변환하는 방법으로서 z순서 색인 기법[11], DOT 색인 기법[7, 14] 등이 여기에 속한다. 세 번째 방법은 k차원의 공간을 분할하는 방식으로 R-tree[1], R+-tree, R*-tree[2], cell tree[15] 등을 들 수 있다.

DOT 공간 색인 기법[6, 7]은 공간 객체의 MBR 정보를 공간 순서화 곡선을 사용하여 하나의 1차원 값으로 변환한 후 그 값을 검색 키로 갖는 B^+-tree 색인 구조를 구성하는 방법이다. 단계적인 색인 생성 방식은 다음과 같다.

우선 k차원에 존재하는 공간 객체의 MBR 정보를 다음 2단계의 변환 과정에 의하여 1차원 값으로 변환한다.

(1) k차원의 원공간(initial space)에 존재하는 공간 객체를 표현하는 MBR 정보를 2k차원의 중간 공간(inter-

mediate space)상에 단편화 없이 하나의 점으로 사상한다. 이것을 1차 변환이라고 한다.

(2) 2k차원의 중간 공간에 하나의 점으로 사상된 공간 객체를 공간 순서화 곡선을 이용하여 1차원상의 한 점을 나타내는 값으로 변환하고, 이 값을 x값(x-value)이라고 부른다. 이것을 2차 변환(공간 변환)이라고 한다.

여기에서 2차 변환에 적용되는 공간 순서화 곡선으로서 공간 객체 간 근접성을 최대한 보존하는 방식을 채택하는 것이 유리하며, 이는 원공간 상의 공간 객체를 상호 인접성이 유지되도록 디스크 상에 클러스터링하여 저장하는 효과를 가진다[16, 17, 18]. 다음 각 공간 객체의 x값을 이용하여 이 값을 검색 키로 갖는 B^+-tree 구조의 색인을 구성하며, 이와 같이 얻어진 B^+-tree 색인 구조를 DOT 공간 색인이라 부른다.

3. 1차원 공간 객체의 영역 질의 처리

3.1 1차원 공간 객체를 위한 DOT 영역 질의 알고리즘

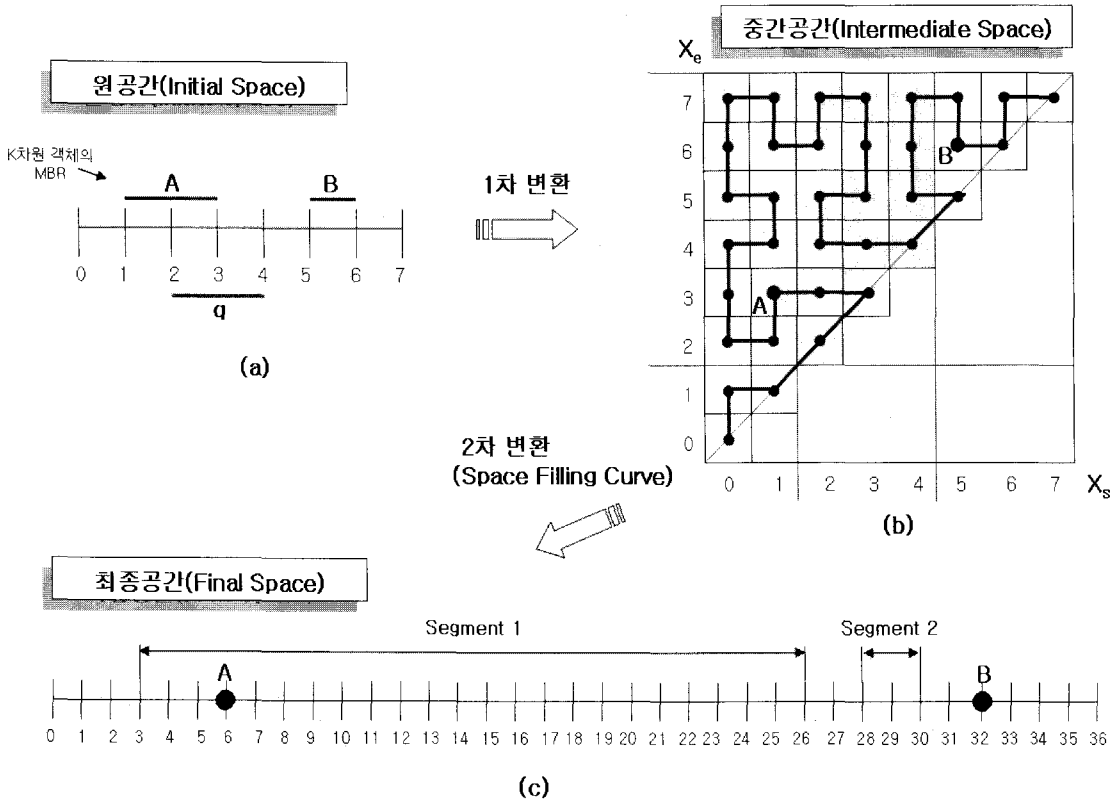
영역 질의는 주어진 질의 영역과 교차하는 모든 공간 객체를 검색하는 연산이다. DOT 공간 색인을 이용한 영역 질의 과정을 그림 1의 예를 들어 간단히 설명하면 다음과 같다. 그림 1-(a)는 1차원의 원 공간에서 공간 객체 A와 B 그리고 질의 영역을 나타내는 q를 표현하고 있다. 다음의 그림 1-(b)는 1차 변환 결과에 의한 공간 객체와 질의 영역을 나타낸다. 1차원 원 공간 상에 존재하는 각 공간 객체의 MBR 정보는 객체의 시작점이 X_s축의 좌표로, 끝점이 X_e축의 좌표로 이용되어 중간 공간인 2차원 공간상의 하나의 점으로 표현되며, 공간 객체 A와 B는 각각 (1,3), (5,6)의 점으로 사상된다. 이 때 1차원 객체의 끝점은 항상 시작점보다 크거나 같은 값을 가지므로 모든 공간 객체는 중간 공간의 대각선을 포함한 위쪽 직각 삼각형에 사상되는 것을 알 수 있다.

또한 DOT 공간 색인 기법에서 질의 영역은 원 공간이 1차원인 경우 중간 공간상에 면적으로 표현된다. 시작점을 q_s로, 끝점을 q_e로 갖는 질의 영역 q는 $X_s \leq q_e$ 와 $X_e \geq q_s$ 의 조건을 만족하는 영역으로 표현 되며, 그림 1-(b)의 음영 처리된 부분이 그 영역을 나타내고 있다. 이 때 공간 객체가 존재하지 않는 대각선 아래 부분은 사용되지 않음을 알 수 있다.

다음의 그림 1-(c)는 2차 변환(공간 변환)에 의하여 최종 공간인 1차원 공간에 사상된 각 공간 객체와 질의 영역을 나타낸다. 그림 1-(b)의 2차원 중간 공간에 대하여 tri-Hilbert 순서화 곡선[6]을 적용하여 공간 객체 및 질의 영역에 대한 1차원의 x값을 추출하면, 공간 객체는 각각 1차원 공간상의 하나의 값(A는 6, B는 32의 x값을 가짐)으로 변환되며, 질의 영역 q는 1차원 공간상의 일련의 점의 집합인 라인 세그먼트의 집합(segment1[3,26]과 segment2[28,30] 상의 x값을 가짐)으로 변환된다.

DOT 색인 기법에서는 이와 같이 얻어진 각 공간 객체의 x값을 이용하여, 그 값을 검색 키로 갖는 B^+-tree 구조의 색인을 구성한다. 따라서 DOT 공간 색인을 이용한 영역 질의는 질의 영역 q의 라인 세그먼트의 집합이 나타내는 범위의 검색키를 가지는 공간 객체를 B^+-tree 구조의 DOT 색인 상에서 찾아내는 과정으로 설명될 수 있다. 즉 그림 1의 영역 질의는 DOT 색인 상에서 탐색 키의 값이 3에서 26사이 또는 28에서 30사이의 객체를 찾는 것에 해당하며, 결국 탐색 키의 값이 6인 객체 A가 검색된다.

DOT 색인을 이용한 영역 질의 알고리즘을 [알고리즘 1]에 보인다. 알고리즘 Range_Query



<그림 1> DOT 색인 기법의 영역 질의 과정

는 원 공간의 질의 영역인 (q_s, q_e)와 중간 공간의 한 변의 크기(grid size) n을 입력으로 받아 다음과 같이 영역 질의를 수행한다. 우선, 함수 Transform_Query_ Range_To_ Line_ Segments()를 호출하여 질의 영역을 1차원 상의 라인 세그먼트들로 변환하고, 이들 값을 LS에 저장한다(line 1). 다음, 함수 Get_ Objects_Using_BTree_ Index()를 호출하여 LS에 해당하는 x값을 가지는 공간 객체를 B^+-tree 구조의 DOT 색인을 이용하여 검색하여 그 결과를 QR에 저장, 반환한다(line 2~3).

[함수 1.1]에 보인 함수 Transform_ Query_Range_To_Line_Segments()는 질의 영역에 해당하는 중간 공간상의 점들을 1차원 x값으로 변환하여 라인 세그먼트의 집합 LS를 구성한다. 여기에서 사용된 함수 Tri_Hilbert()는 tri-Hilbert 공간 순서화 곡선의 운행 경로에 따라 중간 공간의 점을 최종 공간의 값으로 변환하는 공간 변환 연산 함수이다. 또한 Concatenate_Adjacent_ Xvalue(SORT(LS))는 라인 세그먼트를 구성하는 함수로서 질의 영역의 모든 점들에 대하여 x값으로 변환된 결과를 인접한 점들이 연속되는 구간별로 구분하고 시점과 종점을 판단하여 라인 세그먼트를 구성하는 처리를 수행한다. Get_Objects_Using_BTree_Index는 기존의 B^+-tree 색인을 이용한 영역 질의 과정을 나타내므로 자세한 기술을 생략한다.

알고리즘 1. DOT 영역 질의 알고리즘

```
Algorithm Range_Query(qs, qe, n)
1   LS := Transform_Query_Range_To_Line_Segments(qs, qe, n);
2   QR := Get_Objects_Using_BTree_Index(LS);
3   Return QR;
End Algorithm
```

함수 1.1 질의 영역의 공간 변환 함수

```
Function Transform_Query_Range_To_Line_Segments(qs, qe, n)
1   LS := Φ
2   for Xs := 0; Xs < n-1; Xs++ do
3       for Xe := Xs; Xe < n-1; Xe++ do
4           if Xs <= qe and Xe >= qs then
5             { LS := LS ∪ Tri_Hilbert(Xs, Xe); }
6   LS := Concatenate_Adjacent_Xvalue(SORT(LS));
7   Return LS;
End Function
```

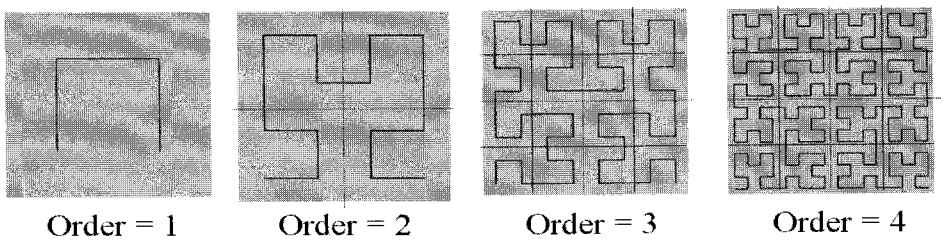
[알고리즘 1]이 좋은 성능을 갖기 위해서는 질의 영역을 최종 공간상의 라인 세그먼트의 집합으로 변환하는 함수 Transform_Query_Range_To_Line_Segments()의 연산 비용을 최소화 시켜야 한다. 그 이유는 질의 영역의 크기에 비례하여 공간 변환 연산을 수행하는 함수 Tri_Hilbert()의 실행 횟수가 증가하기 때문이다. 일반적으로 질의 영역은 중간 공간상에 면적으로 표현되는데, 공간 순서화 곡선의 운행은 이 면적 내에서 반드시 연속적이라 단정할 수는 없다. 따라서 이 면적에 해당하는 모든 점들에 대해 공간 변환 연산을 적용해야만 최종 공간에서 연속적인 라인 세그먼트의 집합을 구성할 수 있다.

질의 영역 (q_s, q_e)의 q_s를 X_s축의 좌표로, q_e를 X_e축의 좌표로 이용하여 중간 공간의 한 점으로 사상한 점을 q라 하였을 때, 이 점 q의 위치에 따라 질의 영역의 크기가 달라진다. 즉, 질의 영역에 해당하는 면적내의 모든 점들에 대해 공간 변환 연산을 적용하는 경우 점 q의 위치에 따른 변환 연산의 횟수를 Nt라고 정의하면 Nt는 $[n \leq Nt \leq (n^2+n)/2]$의 크기를 가지며, 이는 최악의 경우 $O(n^2)$에 해당하므로 DOT 영역 질의 알고리즘의 성능을 저하시키는 중요한 요인으로 작용하게 된다.

3.2 쿼터 분할 기법을 이용한 질의 영역 공간 변환 연산

질의 영역의 모든 점들에 대하여 공간 변환 연산을 적용하는 대신, 질의 영역을 공간 순서화 곡선이 연속 운행되는 부분 영역의 집합으로 분할할 수 있다면 각 분할된 영역의 크기를 이용하여 공간 변환 연산의 비용을 절감할 수 있다. 쿼터 분할 기법은 한 변의 길이가 n인 중간 공간을 한 변의 길이가 n/2인 정방사각형의 쿼터로 재귀적으로 분할하며, 공간 순서화 곡선의 운행 경로를 분석하여 곡선의 운행이 연속되는 쿼터가 질의 영역에 포함되는지를 판별하여, 질의 영역에 포함되는 경우에는 그 시작점에 대하여 쿼터 당 1회의 공간 변환 연산을 적용하여 최종 공간에서의 라인 세그먼트를 찾아내는 방법이다.

공간 순서화 곡선으로서 Hilbert 곡선을 사용하는 경우에 대한 쿼터 분할 기법을 설명하면 다음과 같다. 그림 2에 각 차수



<그림 2> 차수(order)의 증가에 따른 쿼터 분할과 Hilbert 곡선의 운행 방식

(order)에 따른 Hilbert 곡선의 운행 방식을 보인다. 그림과 같이 Hilbert 곡선은 크기가 n인 중간 공간을 한 변의 길이가 n/2인 쿼터로 재귀적으로 분할하였을 때 각 쿼터 내에서 연속적인 값을 갖도록 운행하는 성질을 가지고 있다. 따라서 이러한 성질을 이용하면 분할되는 쿼터로부터 최종 공간에서의 연속적인 x값의 범위를 예측할 수 있다. 이와 함께 고려되어야 할 것은 Hilbert 곡선이 분할된 쿼터의 어느 점에서 시작되는지를 판단하는 것이다. 이는 각 쿼터 내에서의 곡선의 시작점의 x값을 알게 되면 라인 세그먼트의 범위는 시작점의 x값에 쿼터의 한 변의 길이의 제곱을 합하고 1을 빼는 것으로 쉽게 계산될 수 있기 때문이다.

Hilbert 곡선의 운행 규칙은 다음과 같이 요약될 수 있다. 여기에서는 쿼터의 각 사분면을 그림 3-(a)와 같이 오른쪽 위의 사분면부터 시계 방향으로 각각 1 사분면, 2 사분면, 3 사분면, 4 사분면으로 부른다.

(1) Hilbert 곡선은 차수(Order)가 1인 경우, 중간 공간을 3-4-1-2 사분면의 순서로 운행한다. 이것을 Pattern3412의 패턴이라고 부른다. 즉, 중간 공간은 처음에 Order=1인 쿼터로 간주되며 운행 순서는 ⌐↓와 같이 표현된다. 차수가 2인 경우, 분할된 각 쿼터에서의 Hilbert 곡선의 운행 패턴을 그림 3-(b)에 보인다.

(2) 쿼터의 재귀적인 분할에 대한 Hilbert 곡선의 운행 패턴은 표 1과 같이 일정한 규칙을 가진다. Hilbert 곡선은 Pattern3412, Pattern3214, Pattern1432, Pattern1234의 4가지 패턴으로 이루어져 있으며, 쿼터가 분할됨에 따라 각 사분면의 쿼터는 각각 표 1과 같이 정해진 운행 패턴을 가지게 된다. 표 1을 이용하여 쿼터의 다음 분할에 따른 각 사분면 쿼터에서의 곡선의 운행 패턴과 시작점을 쉽게 찾을 수 있다.

(a) (b)

<그림 3> 쿼터의 사분면 정의와 Hilbert 곡선의 초기 운행 방식

<표 1> Hilbert 곡선의 운행 규칙

Order	Order + 1			
	1 사분면쿼터	2 사분면 쿼터	3 사분면 쿼터	4 사분면 쿼터
Pattern3412	Pattern3412	Pattern1432	Pattern3214	Pattern3412
Pattern3214	Pattern3214	Pattern3214	Pattern3412	Pattern1234
Pattern1432	Pattern1234	Pattern3412	Pattern1432	Pattern1432
Pattern1234	Pattern1432	Pattern1234	Pattern1234	Pattern3214

다음의 [함수 1.2]는 쿼터 분할 기법을 적용한 질의 영역의 공간 변환 함수를 나타낸다. 우선, Initialize_Quarter()는 쿼터의 크기 및 관련 정보를 초기화 하는 작업을 수행하며, Hilbert 곡선의 초기 운행 방향인 Pattern3412를 초기 운행 패턴으로 지정한다(line 2). 다음으로, 함수 Split_Query_Range()는 쿼터 Q를 재귀적인 서브 쿼터로 분할하면서 분할된 쿼터가 질의 영역에 속하는 경우에는 쿼터의 운행 패턴을 기준으로 공간 순서화 곡선의 시작 위치를 찾아 단 한 번의 공간 변환 연산으로 쿼터 크기

함수 1.2 쿼터 분할 기법을 이용한 질의 영역의 공간 변환 함수

```
Function Transform_Query_Range_To_Line_Segments(qs, qe, n)
1   LS := Φ
2   Q := Initialize_Quarter(qs, qe, n);
3   Call Split_Query_Range(Q, LS, qs, qe);
4   Return LS;
End Function

Function Split_Query_Range(Q, LS, qs, qe)
5  if Is_In_Region(Q, qs, qe) Then
6    { LSnew.Start := Tri_Hilbert(Start_Point(Q));
7      LSnew.End := LSnew.Start + Q.size^2 - 1;  LS := LS ∪ LSnew; }
8  else
9    if Q.size > 1 then
10     {  Qsub.size = Q.size / 2;
11        for SubQuarterSeq := 1; SubQuarterSeq ≤ 4; SubQuarterSeq++ do
12          {  Qsub.pattern := next_pattern(Q, SubQuarterSeq);
13             Qsub.upperleft := next_upperleft(Q, SubQuarterSeq);
14             Call Split_Query_Range(Qsub, LS, qs, qe); }}
End Function

Function Is_In_Region(Q, qs, qe)
15  if In_UpperDiagonal(Q.UpperLeft) and In_Query_Region(Q.UpperLeft, qs, qe) then
16    { return TRUE; }
17  else
18    { return FALSE; }
End Function
```

만큼의 연속된 라인 세그먼트를 반환한다. 한편, 분할된 쿼터가 질의 영역에 속하지 않는 경우에는 이것을 다시 4개의 서브 쿼터로 나누어 각 서브 쿼터에 대하여 공간 순서화 곡선의 운행 경로와 서브 쿼터의 위치를 설정하여 재귀적으로 Split_Query_Range()를 호출한다(line 3). 함수 Is_In_Region(Q, q_s, q_e)는 분할된 쿼터 Q가 (q_s, q_e)로 설정된 질의 영역에 속하는 지를 판별하는 함수이다. 이 함수는 분할된 쿼터가 중간 공간의 대각선 위쪽에 위치하면서 설정된 질의 영역에 완전히 포함되는 경우 TRUE 값을 반환한다(line15~18)

[함수 1.2]의 함수 Transform_Query_Range_To_Line_Segments()에서는 분할된 쿼터마다 한 번의 공간 변환 연산을 수행한다. 즉, 쿼터 분할 기법을 적용한 공간 변환 연산의 횟수는 분할되는 쿼터의 개수와 같으며, 따라서 분할되는 쿼터의 수가 알고리즘의 성능을 좌우하는 중요한 요소가 된다. 중간 공간의 한 변의 크기를 n이라고 하였을 때, 분할되는 쿼터의 개수는 질의 영역을 중간 공간에 사상한 점 q에 의한 질의 영역의 범위에 따라 다르다. 따라서 [함수 1.2]의 공간 변환 연산의 횟수 Nt는 $n \leq Nt \leq 2n-1$로 계산되어 [함수 1.1]과 비교하여 향상된 성능을 보인다. 즉 최악의 경우, 공간 변환 연산의 복잡도가 $O(n^2)$에서 $O(n)$으로 개선될 수 있음을 알 수 있다.

3.3 삼각형 쿼터(Tri-Quarter) 분할을 이용한 추가 성능 개선

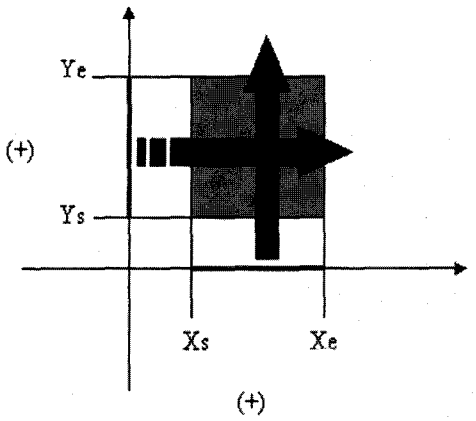
쿼터 분할 기법에서는 질의 영역에 대하여 정방사각형의 쿼터 분할을 재귀적으로 수행함으로써 공간 변환 연산의 비용을 절감시킨다. DOT 색인 기법에서는 모든 공간 객체와 질의 영역이 대각선 위쪽 삼각형 부분에만 적용되므로 공간 순서화 곡선도 그림 1의 경우처럼 대각선 위쪽 삼각형만을 운행하는 tri-Hilbert 곡선을 사용할 수 있다. 따라서 쿼터의 분할 과정에 있어 대각선에 걸쳐지는 쿼터는 반드시 사각형이 아니라 대각선을 빗변으로 하는 직각삼각형의 경우도 연속되는 쿼터라고 가정할 수 있다. 이 경우, 단 1회의 공간 변환 연산에 의하여

함수 1.3 삼각형 쿼터 분할 기법을 추가한 질의 영역의 공간 변환 함수

```
Function Split_Query_Range(Q, LS, qs, qe)
1    if Is_In_Region(Q, qs, qe) then
2       { LSnew.Start := Tri_Hilbert(Start_Point(Q));
3          if Q.TriQuarter = TRUE then
4             { LSnew.End := LSnew.Start + (Q.size^2 + Q.size) /2 - 1; }
5          else
6             { LSnew.End := LSnew.Start + Q.size^2 - 1; }
7          LS := LS ∪ LSnew }
8    else
9       if Q.size > 1 then
10         {  Qsub.size = Q.size / 2;
11            for SubQuarterSeq := 1; SubQuarterSeq ≤ 4; SubQuarterSeq++ do
12            {  Qsub.pattern := next_pattern(Q, SubQuarterSeq);
13               Qsub.upperleft := next_upperleft(Q, SubQuarterSeq);
14               Call Split_Query_Range(Qsub, LS, qs, qe); }}
End Function
```

삼각형의 면적으로 라인 세그먼트를 계산한다면 질의 영역의 대각선 경계 부분에서 보다 작은 크기의 쿼터(최소크기 = 1)로 분할되는 경우를 최적화 할 수 있어 추가로 연산 비용을 절감할 수 있게 된다.

다음의 [함수 1.3]은 삼각형 쿼터 분할을 적용하여 [함수 1.2]를 추가로 개선한 함수이다. 함수 Is_In_Region(Q, q_s, q_e)는 삼각형 쿼터를 판별하도록 수정되었다. 또한 삼각형 쿼터의 라인 세그먼트 크기를 계산하는 방법은 $(Q.size^2 + Q.size)/2 - 1$ 과 같다(line 4).



<그림 4> 2차원 공간 객체의 특징

4. 2차원 공간 객체의 영역 질의 처리

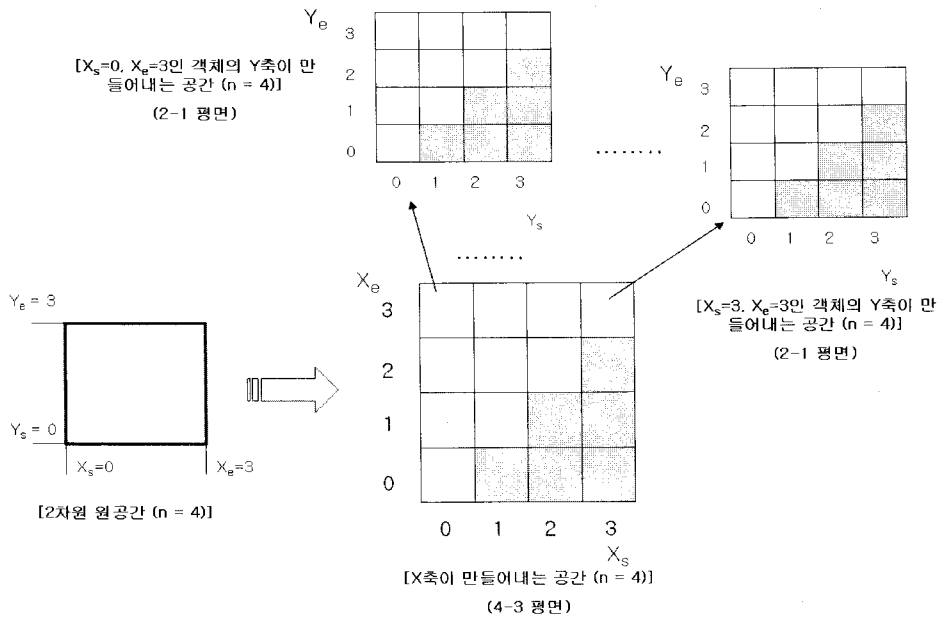
k차원의 원 공간 객체에 대하여 DOT 공간 색인 기법을 적용하면 2k차원의 중간 공간을 고려 대상으로 하여야 한다. 2차원 원 공간 객체는 4차원 중간 공간을 가지게 되는데, 4 차원 이상의 공간 모습은 직관적으로 묘사하기 어렵다. 본 장에서는 4차원 중간 공간의 모습을 해석하고, 공간 순서화 곡선의 운행을 분석하기 위하여 4차원 중간 공간 해석 모델을 제시하고, 제시된 모델이 DOT 공간 색인 기법을 적용하기에 적합함을 보인다.

4.1 4차원 중간 공간 해석 모델

2차원 공간 객체는 그림 4와 같이 X축과 Y축 모두 양(+)인 공간에 존재하며 $X_s \leq X_e$ 와 $Y_s \leq Y_e$인 공간으로 표현되고, 시간의 차원 t = 0 인 관점에서 X축은 시작점 X_s로부터 끝점 X_e인 선분이 Y축 방향으로, Y축은 시작점 Y_s로부터 끝점 Y_e인 선분이 X축 방향으로 평행 이동 하면서 서로 교차하는 영역으로 설명될 수 있다.

그림 4가 표현하는 특징을 이용하여 X축과 Y축에 대하여 X_s, X_e, Y_s, Y_e의 네 개의 좌표를 표현하는 파라미터를 설정한다. 이때 X축과 Y축의 인접성을 유지하는 파라미터의 조합은 4 가지가 있으며, 이 논문에서는 X_s를 4차원, X_e를 3차원, Y_s를 2차원, Y_e를 1차원으로 표현하는 파라미터로 사용한다(그림 5 참조). 이렇게 표현된 4차원 중간 공간의 성질은 다음과 같이 정리된다. 2차원 공간 객체의 X축은 제 4차원 좌표축과 제 3차원 좌표축이 각각 양(+)의 방향으로 교차하며 이루는 평면(이를 4-3평면이라 함)상에 한 개의 단위 셀로 표현되며, $X_s \leq X_e$인 관계가 성립하므로 1차원 공간 객체의 중간 공간과 마찬가지로 대각선 위쪽 삼각형 부분에만 사상 된다. X축과 Y축은 서로 직교하므로 4-3평면의 한 단위 셀은 점이 아니라 Y축으로 표현되는 2차원 공간을 소유하는 3차원 공간의 일부가 된다. Y축 역시 제 2차원 좌표축과 제 1차원 좌표축이 각각 양의 방향으로 교차하며 이루는 평면(이를 2-1평면이라 함)상에 한 점으로 표시된다.

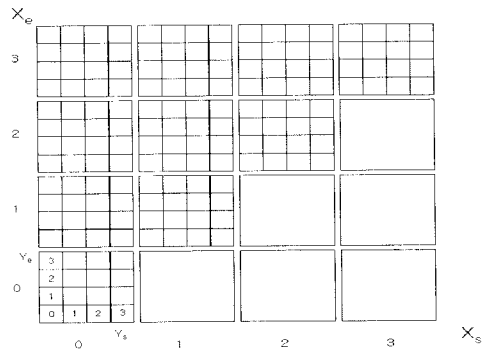
결국 2차원 공간 객체는 (X_s, X_e, Y_s, Y_e) 좌표로 표현된 4차원 중간 공간 상의 한 점으로 표현될 수 있다. 다른 평면과의 관계,



<그림 5> (Xs, Xe, Ys, Ye) 좌표를 이용한 4차원 중간 공간 표현의 예 (n=4의 경우)

즉 3-2평면(X_e, Y_s좌표가 만들어내는)과 같은 또 다른 평면들이 어떤 모양으로든 4-3평면과 2-1평면에 교차 또는 인접해 있다고 가정해 볼 수 있지만, 그 모습을 기하학적으로 추정하기는 곤란할 뿐만 아니라 그림 4에서 설명한 2차원 공간 객체의 특징으로 볼 때 실재 공간이 아니므로 고려할 필요가 없다. 그림 5은 (X_s, X_e, Y_s, Y_e) 좌표를 이용하여 4차원으로 변환된 중간 공간의 성질을 보여준다. 여기서 대각선을 포함한 위쪽 삼각형 부분에만 공간 객체 및 질의 영역이 사상되므로 대각선 아래쪽 삼각형 부분은 음영 처리하여 표현하였다.

실제로 4차원 공간에서 평면간의 인접 또는 교차되는 모습이나 서로 직교하는 4개의 좌표축을 가시적으로 묘사할 수 없다. 그러나 4번째 차원의 축을 시간이라고 가정하면 3차원 공간을 동일 평면에 시간 순서로 나열해 볼 수 있을 것이다. 결국 X_s 좌표 값을 시간 순서로 정한다면 4차원 중간 공간의 모습은 그림 6에 제시된 모양으로 3차원 공간들의 집합으로 묘사할 수 있게 된다. 이 그림에서 X_s, X_e 좌표가 직교하며 만들어내는 공간(4-3평면)의 셀들은 Y_s, Y_e로 만들어지는 평면(2-1평면)을 하나씩 소유하고 있는 모습으로 표현되어 있으며 각 평면에서 대각선 아랫부분 삼각형은 사용되지 않으므로 음영 처리로 표현하였다.

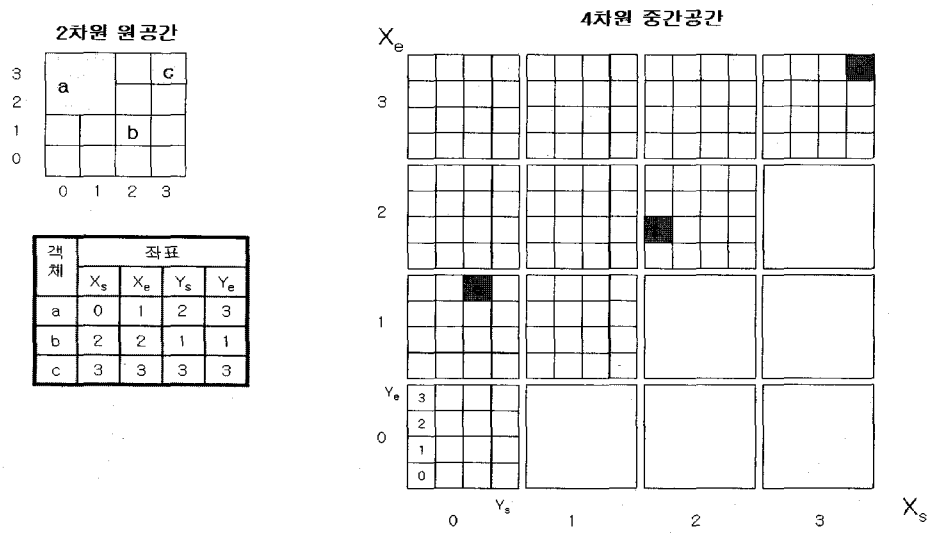


<그림 6> n=4인 4차원 중간 공간 해석 모델

4.2 해석 모델을 이용한 DOT 색인 기법 적용

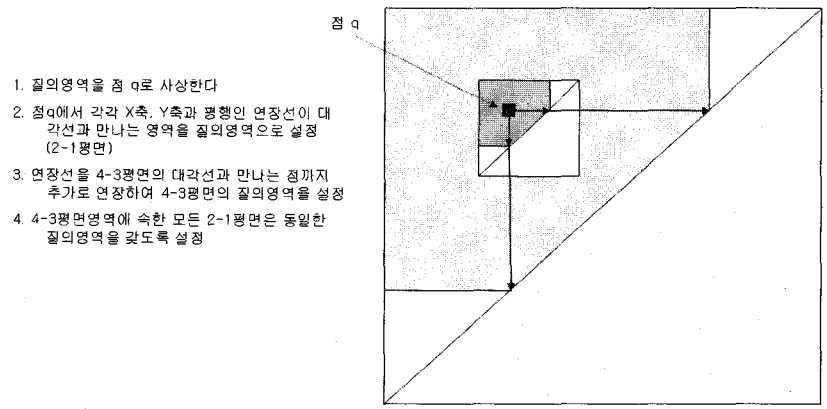
본 절에서는 4.1절에서 제시한 4차원 중간 공간 해석 모델이 DOT 색인 기법을 적용하기에 적합함을 보인다. 다음의 그림 7은 2차원 공간 객체의 1차 변환 과정을 보인다. 원 공간에 존재하는 객체 a, b, c는 각각 (X_s, X_e, Y_s, Y_e)인 좌표가 (0,1,2,3), (2,2,1,1), (3,3,3,3)이며 이들은 해석 모델 상에 한 점으로 사상된다.

4차원 중간 공간 상에서의 질의 영역의변환은 1차원의 경우를 확장하여 설정할 수 있다. 그림 8에 점 q로 사상되는 질의 영역을 4차원 중간 공간에 설정하는 도식적 과정을 보인다. 그림 9의 구체적 예를 들어 설

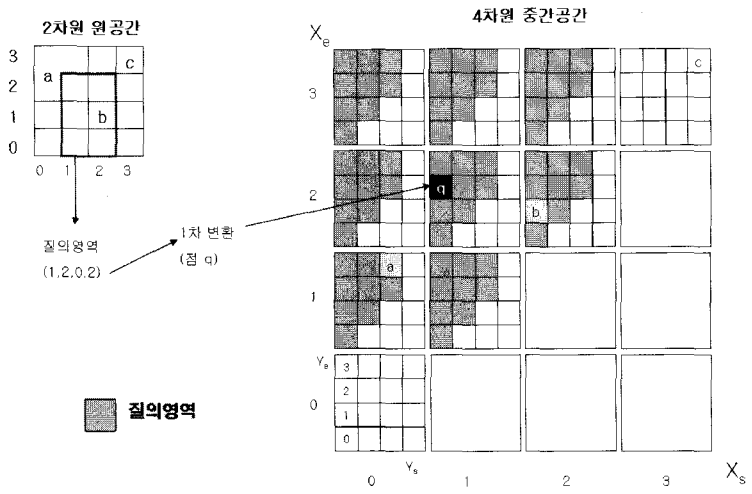


객체	좌표			
	X_s	X_e	Y_s	Y_e
a	0	1	2	3
b	2	2	1	1
c	3	3	3	3

<그림 7> 2차원 공간 객체의 1차 변환

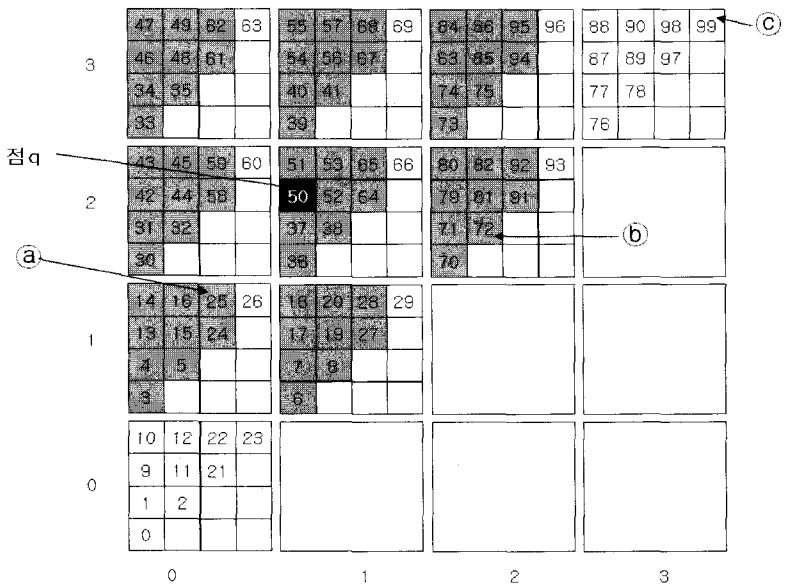


<그림 8> 질의 영역의 변환 방법



<그림 9> 4차원 중간 공간 해석 모델에 설정된 질의 영역

명하면 질의 영역은 다음과 같이 변환된다. 먼저 그림 9에 (1,2,0,2)로 표현된 질의 영역을 중간 공간에 한 점 q로 사상한다. 다음은 그림 9의 화살표와 같이 2-1평면에 대하여 점 q를 1차원의 경우처럼 X축(+방향)과 Y축(-방향)으로 연장하여 그 연장선이 대각선과 만나는 점을 기준으로 질의 영역을 설정하고, 다시 4-3평면으로 연장하여 4-3평면의 대각선에 만나는 점을 기준으로 4-3평면 질의 영역을 정하되 영역에 속하는 모



<그림 10> 4차원 중간 공간 상의 2차 변환 예

든 2-1평면들이 처음 설정한 2-1평면과 동일한 질의 영역을 가지도록 설정한다. 이러한 질의 영역은 $QX_s \leq X_e$ and $QX_e \geq X_s$ and $QY_s \leq Y_e$ and $QY_e \geq Y_s$로 정의할 수 있다.

DOT 공간 색인 기법을 적용하기 위해서는 1차 변환된 공간 객체와 질의 영역을 1차원의 x값을 가지는 최종 공간으로 변환하여야 한다. 1차원의 경우와 같이 4차원 중간 공간을 1차원의 값으로 변환시킬 수 있는 공간 순서화 곡선을 사용한다. 본 논문에서는 대각선 위쪽만을 운행하는 4차원 tri-Peano 곡선을 사용한다. Peano 곡선은 Hilbert 곡선에 비해 비교적 간단한 경로의 운행을 하며, 쿼터 내에서는 곡선의 시작점이 항상 좌하점으로 일정하여 4차원 중간 공간을 운행하는 경로와 방향을 분석하고, 예측하기에 용이한 장점이 있다. 다음의 그림 10은 그림 9에 제시된 2차원 공간 객체와 질의 영역(음영처리 된 부분)을 4차원 중간 공간에 사상한 다음 tri-Peano 곡선을 적용하여 중간 공간의 각 셀들에 대해 대응되는 최종 공간에서의 1차원 x값을 표현한 예를 보인다. 공간 객체는 각각 a=25, b=72, c=99의 x값을 가지며, 음영 처리된 질의 영역에 각각 1차원 x값이 할당됨을 볼 수 있다.

4.3 DOT 색인 기법을 이용한 2차원 공간 객체의 영역 질의

2차원 공간 객체의 영역 질의는 1차원 공간 객체의 경우와 동일하게 적용할 수 있다. 질의 영역을 최종 공간상의 1차원 라인 세그먼트의 집합으로 변환하고, 최종 공간에서 라인세그먼트의 집합이 나타내는 범위에 속하는 공간 객체를 DOT 색인을 통하여 검색할 수 있다. 결국 공간 순서화 곡선이 4차원 곡선이며, 4차원 중간 공간에 질의 영역이 설정되는 것을 제외하면 1차원의 경우와 동일하게 적용할 수 있고 주색인 기법이 가지는 클러스터링 효과와 디스크 액세스면에서의 장점을 활용할 수 있게 된다.

2차원 공간 객체를 위한 영역 질의 방식은 제 3장에서 보인 [알고리즘 1]과 근본적으로 동일하다. 단, 원 공간의 질의 영역이 (QX_s, QX_e, QY_s, QY_e)로 확장되어야 한다. 또한 질의 영역의 공간 변환 함수를 [함수 2.1]과 같이 확장한다. 이때 공간 순서화 곡선은 대각선 위쪽만을 운행하는 4차원 Tri_Peano곡선을 적용하였으므로 [함수 1.2]의 경우와는 달리 공간순서화 곡선의 시작점은 항상 쿼터의 좌하점이다. 먼저 4-3평면의 질의 영역을 쿼터 분할하여(line 3) 분할 된 쿼터가 질의영역에 속하는 경우 4-3평면의 모든 셀들에 대하여 다시 2-1평면을 쿼터 분할하여(line 6 ~ 8) 라인 세그먼트를 얻는다. 이때 Split_Query_Range21()함수는 2-1평면을 쿼터 분할하는 함수로써 [함수 1.2]에서 제시된 함수와 동일하여 자세한 기술을 생략한다. 단, 2-1평면 쿼터가 질의 영역에 속하는 경우 라인 세그먼트의 크기는 그림 11과 같이 계산된다.

[함수 2.1]에 제시된 방식은 1차원의 경우를 단순 확장한 것으로써, 2차 변환에 소요되는 연산 비용이 매우 크다. 그 이유는 질의 영역에 속하는 모든 4-3평면의 셀마다 2-1평면을 분할하여 라인 세그먼트를 계산하기 때문이다. 1차원의 경우처럼 2-1평면의 연산 비용을 2n-1로 간주할 때, 최소 $2n^2 - n$ 에서 최대 $(2n^3 + n^2 - n)/2$ 가 되므로 $O(n^3)$의 복잡도를 가지므로 만족할 만한 알고리즘의 성능을 기대하기 어렵다. 또한 이 방법은 2차원 중간 공간의 경우와 달리 4차원의 중간 공간에 설정된 2-1평면을 분할할 때 공간순서화 곡선의 운행이 연속되는 영역이 다르므로 추가적인 연산 비용이 소요된다.

함수 2.1 2차원 공간 객체를 위한 질의 영역의 공간 변환 함수

```
Function Transform_Query_Range_To_Line_Segments(QXs, QXe, QYs QYe, n)
1   LS := Φ
2   Q43 := initialized_Quarter(QXs, QXe, n);
3   Call Split_Query_Range(Q43, LS);
4   Return LS;
End Function

Function Split_Query_Range(Q43:Quarter, LS:Line Segments)
5   if Is_In_Region(Q43) Then
6       for each cell C43 in Q43 do
7           { Q21 := initialized_Quarter(QYs, QYe, n) LS21 := Φ
8             Call Split_Query_Range21(C43, Q21, LS21) LS := LS ∪ LS21}
9   else
10      if Q43.size > 1 then
11          {  Q43sub.size = Q.size / 2;
12             for SubQuarterSeq := 1; SubQuarterSeq < 4; SubQuarterSeq++ do
13                 {  Q43sub.upperleft := next_upperleft(Q43, SubQuarterSeq);
14                    Call Split_Query_Range(Q43sub, LS); }}
End Function
```

4.4 질의 영역의 특징을 이용한 성능 개선 방법

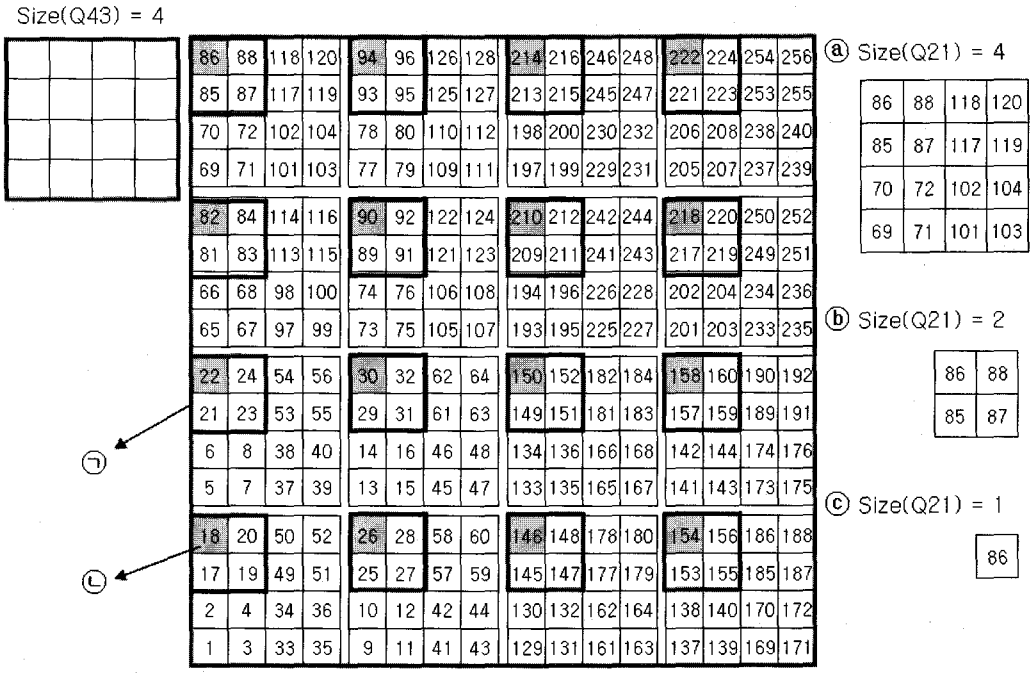
4.2절에서 질의 영역에 속하는 모든 2-1평면의 질의 영역이 모두 동일하다는 것을 밝힌 바 있다. 이것은 2차 변환 연산 비용을 절감할 수 있는 중요한 특징 중 하나이다. 직관적으로 질의 영역이 설정된 2-1평면을 단 1회만 쿼터 분할하여 그 정보를 메모리에 유지하고, 4-3평면을 쿼터로 분할하면서 질의 영역에 속하는 4-3평면 쿼터의 시작점을 이용하여 상대적인 위치를 계산하면 4-3평면쿼터의 크기와 2-1평면쿼터의 크기를 이용하여 4-3평면 쿼터 당 단 1회의 변환 연산으로 라인 세그먼트를 산출할 수 있게 된다. 이 경우 알고리즘의 복잡도는 $(2n - 1)^2$ 이므로 연산 비용은 $O(4n^2 - 4n + 1)$로 절감될 수 있다. 그 이유는 4차원 중간 공간 해석 모델을 2차원 배열(2-1평면)을 소유하는 2차원 배열(4-3평면)로 제시하였으므로 원 공간의 각 차원당 2n-1개의 쿼터로 분할하여 라인 세그먼트를 계산할 수 있다고 판단되기 때문이다.

그러나, 이 방법은 하나의 4-3평면 쿼터 내에서는 이미 분할 해 놓은 2-1평면의 쿼터가 모두 동일 위치에 있는 경우라면 공간

```
{ LS21new.Start := Tri_Peano_Value(X1(C43), X2(C43), Y1(Q21), Y2(Q21));
  LS21new.End := LS21new.Start + Q21.size^2 - 1;
  LS21 := LS21 ∪ LS21new
}

X1(C43) : 4-3평면쿼터에 속한 한 셀의 Xs 축 좌표
X2(C43) : 4-3평면쿼터에 속한 한 셀의 Xe 축 좌표
Y1(Q21) : 2-1평면쿼터의 시작점 Ys 좌표
Y2(Q21) : 2-1평면쿼터의 시작점 Ye 좌표
```

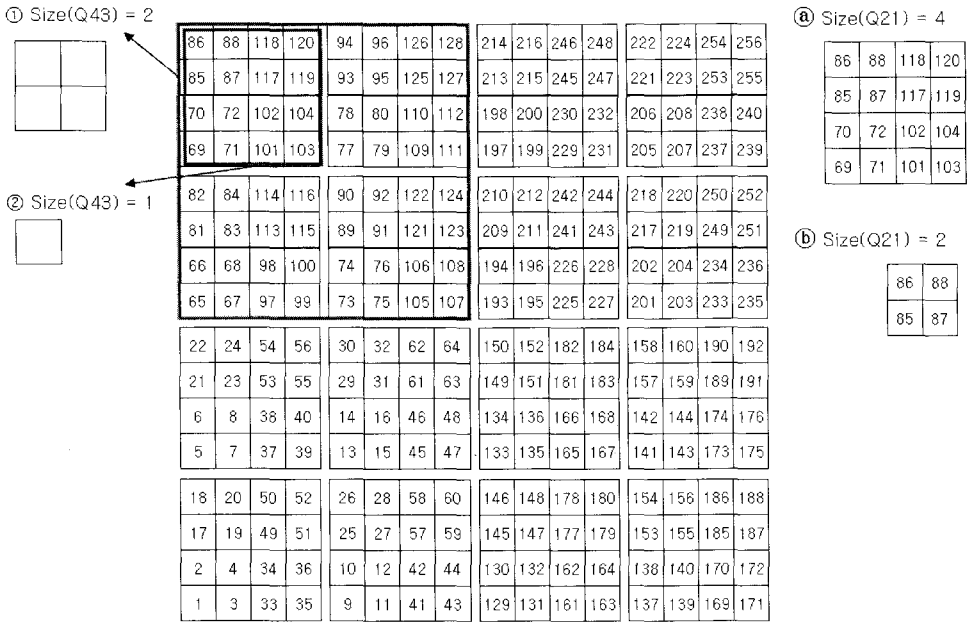
<그림 11> 질의 영역에 속하는 2-1평면 쿼터의 라인 세그먼트 계산



<그림 12> 4차원 쿼터 분할과 평면간 쿼터 크기의 관계(n = 4, Size(Q43) = 4)

순서화 곡선이 연속 운행 한다는 가정이 성립할 경우에 적용할 수 있다. 중간 공간의 차원이 2차원인 경우 가로와 세로의 크기가 동일한 2차원 면적을 쿼터로 간주하였듯이 4차원의 경우 쿼터란 4개의 축이 모두 동일 크기를 가지는 4차원 공간이 쿼터이며, 이 쿼터 내에서만 공간 순서화 곡선이 연속 운행하는 것이 보장된다. 따라서 분할된 4-3평면 쿼터에 속해 있는 모든 2-1평면 쿼터들은 그 크기가 4-3평면 쿼터의 크기와 동일한 쿼터인 경우에는 반드시 모두 연속이다.

그림 12의 예를 들어 설명하면 다음과 같다. 그림 12에서 각 셀에 부여된 번호는 1부터 시작되는 4차원 Peano곡선의 운행 순서를 의미하며, 해석 모델에 기초하여 n=4인 2-1평면이 모두 16개가 배치되어 있다. 설명을 간단하게 하기 위하여 그림 12가 질의 영역에 속해 있는 하나의 분할된 4-3평면 쿼터라고 가정한다. 이 4-3평면 쿼터의 크기가 4 (Size(Q43)=4) 라면 이 영역에 속한 모든 점들이 대상이 된다. 이때 ⓐ의 경우처럼 분할된 2-1평면 쿼터의 크기가 역시 4라면 4-3평면내의 모든 점들을 운행하는 Peano곡선은 연속된다(1~256). 반면 ⓑ의 경우처럼 2-1평면 쿼터의 크기가 2라면 각 2-1평면상의 동일 위치에 있는 점들을 조사하여 보면(그림 12의 ㉠으로 표시된 사각형으로 둘러싸인 쿼터들) 각각 17~32, 81~96, 145~160, 209~224로 4개의 라인 세그먼트로 나뉘어지므로 연속되지 않는다. 또한 ⓒ의 경우처럼 2-1평면 쿼터의 크기가 1인 경우는 모두 길이가 1인 별개의 라인 세그먼트로 간주되어야 한다. 즉 ㉡으로 표시된 음영 처리된 점들로 각각 18, 22, 26, 30, 82, 86, 90, 94, 146, 150, 154, 158, 210, 214, 218, 222로 크기가 1인 별



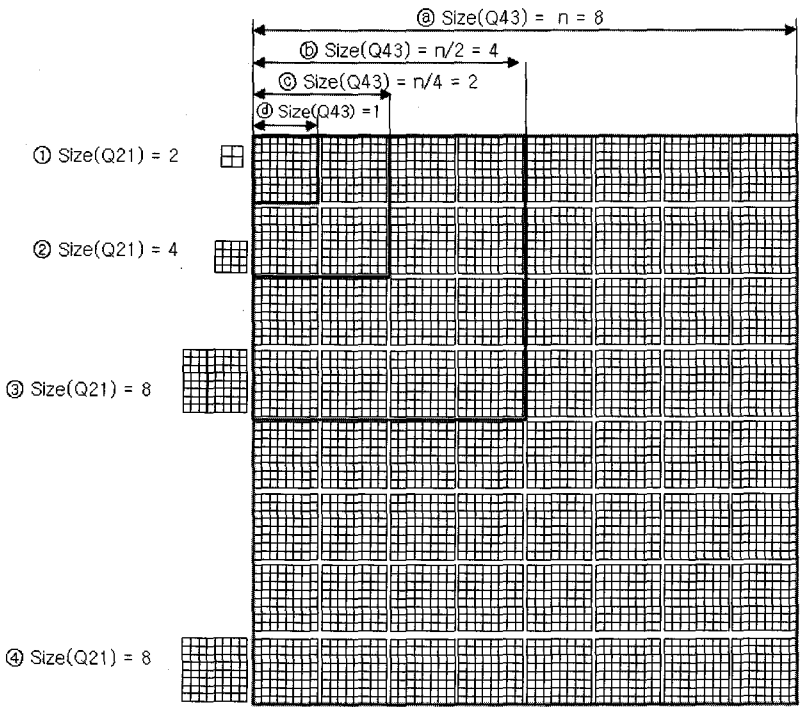
<그림 13> 4차원 쿼터 분할과 평면간 쿼터 크기의 관계
(상위평면크기(Size(Q43) < 하위평면크기(Size(Q21))

개의 라인 세그먼트가 된다.

그림 13은 상위 평면 쿼터인 4-3평면 보다 하위 평면인 2-1평면 쿼터의 크기가 큰 경우 쿼터의 분할을 표현하고 있다.

①의 경우처럼 4-3평면 쿼터의 크기가 2인 경우 그림 13의 바깥쪽 사각형 테두리에 속하는 쿼터가 질의 영역에 속하는 경우 분할된 2-1평면 쿼터의 크기가 ⓐ와 같이 4-3평면 크기보다 큰 4라면 해당되는 모든 점들(65~128)은 연속이다. 물론 ⓑ와 같이 2-1평면 쿼터의 크기가 4-3평면 쿼터와 동일한 크기라면 그림 13의 경우처럼 해당하는 모든 점들도 연속이다(81~96). ②의 경우처럼 4-3평면 쿼터의 크기가 1인 경우 하위 평면의 크기에 따라 연속되는 구간이 다르게 된다. 이때는 하나의 2-1평면을 소유하는 경우이지만 공간 순서화 곡선은 중간 공간 차원의 개수만큼(2차원 원 공간인 경우 4개의 점)만 운행하고 인접한 다른 2-1평면을 운행하기 때문에 ⓐ와 같이 2-1평면 쿼터의 크기가 4인 경우는 각각 69~72, 85~88, 101~104, 117~120의 네 개의 구간으로 분리된다. 그러나 ⓑ와 같이 2-1평면 쿼터의 크기가 2인 경우 연속 운행이 보장되는 최소 크기이므로 연속된다. 그러나 이 경우는 n = 4인 경우를 예시한 것으로써 4-3평면의 크기와 중간 공간의 크기 n과의 관계를 확인하기 위하여 그림 14는 n=8인 경우 평면간 쿼터 크기에 따라 연속 운행이 보장되는 쿼터 크기를 제시하고 있다.

그림 14는 n = 8인 경우 4차원 Peano곡선의 운행을 모두 조사한 결과를 정리한 것으로서 상위 평면의 크기에 따라 공간 순서화 곡선의 연속 운행이 보장되는 하위 평면 쿼터의 최대 크기를 판별할 수 있는 기준을 보인다. ①, ②, ③, ④는 각각 4-3평면의 크기가 1, 2, 4, 8일 때 공간 순서화 곡선의



<그림 14> 상위평면 크기에 따라 연속이 보장되는 하위평면쿼터의 크기 (n=8인 4차원 중간공간)

연속이 보장되는 최대 크기의 2-1평면 쿼터를 나타내고 있으며 그 규칙은 min[(Size(Q43) * 2), n]이 된다. 즉 상위 평면 쿼터의 크기의 두 배와 중간 공간의 크기 중 작은 값을 가지는 크기의 2-1평면 쿼터들의 공간 순서화 곡선이 연속된다는 뜻이다. 중간 공간의 4-3평면이 분할되는 규칙은 ⓐ, ⓑ, ⓒ, ⓓ와 같이 재귀적으로 n/2씩 분할하게 되는데 이 쿼터에 속한 하위 평면(2-1평면)의 크기는 각각 1, 2, 4, 8의 크기로 분할 될 수 있다. 2-1평면의 크기가 1인 경우는 그림 12에서 밝혔듯이 각 점들을 모두 별개의 라인 세그먼트로 취급하여야 하며, 4-3평면의 크기보다 작은 경우는 4-3평면을 2-1평면의 크기로 추가로 분할하여야 한다. 반면 하위 평면 쿼터의 크기가 상위 평면의 두 배의 크기를 초과하는 경우는 연속이 보장되지 않는다. 즉 ⓓ의 경우처럼 Size(Q43) = 1인 경우 Size(Q21)이 4, 8인 경우는 공간 순서화 곡선이 연속되지 않으며, ⓒ의 경우처럼 Size(Q43) = 2인 경우 Size(Q21)이 8이라면 역시 연속되지 않는다. 따라서 이 경우는 2-1평면을 다시 4-3평면의 두 배의 크기를 가지는 쿼터로 추가로 분할하여 라인 세그먼트를 계산하여야 한다.

따라서 그림 12, 13, 14를 통하여 알 수 있듯이 4차원 중간 공간에서 공간 순서화 곡선의 연속 운행이 보장되는 경우를 고려하여 라인 세그먼트를 산출하는 방법을 정리하면 다음과 같다.

(1) Size(Q43) = Size(Q21)인 경우 상위 평면과 하위 평면의 쿼터 크기가 같은 경우이므로 4-3평면에 속한 동일

위치의 2-1평면 쿼터의 모든 점들은 연속되며 그 크기는 $Size(Q43)^2 * Size(Q21)^2 - 1$로 계산된다.

(2) Size(Q43) > Size(Q21)인 경우 상위 평면 쿼터의 크기가 하위 평면 쿼터보다 큰 경우로써 4-3평면을 2-1평면 크기의 쿼터들로 추가로 분할하여 각각을 (1)의 경우와 같이 라인 세그먼트로 계산하여야 한다.

(3) Size(Q21) = 1인 경우 4-3평면 쿼터 내의 모든 2-1평면 쿼터의 점들은 연속이 보장되지 않으므로 4-3평면의 면적만큼 길이가 1인 각각의 라인 세그먼트로 계산한다. 또한 Size(Q43) = 1이면서 Size(Q21) > 2 인 경우는 (3)의 경우를 이용하여 연속이 보장되는 최소 크기인 Size(Q21) = 2 로 추가로 분할 한 뒤 라인 세그먼트를 산출한다.

질의 영역의 공간 변환 연산을 최소화하기 위하여 이 들 사실을 고려하여 [함수 2.1]의 line 6에서 line 8을 다음의 [함수 2.2]와 같이 재 작성하였다. 여기에서 line 1 부터 line 2는 최초 1회만 2-1평면을 분할하여 분할된 쿼터의 집합을 Quarters21에 모리에 유지하는 과정을 나타내며, Get_Line_Segment는 두 쿼터의 크기를 이용하여 최종 공간에서의 라인 세그먼트를 계산하여 반환하는 함수를 나타낸다.

함수 2.2 쿼터의 추가 분할을 고려한 질의 영역의 공간 변환 함수

```
....
1   if First_Quarter_In_Region then
2     { Quarters21 := Split_QueryRange21 First_Quarter_In_Region := false; }
3   for each Quarters Q21 in Quarters21 do
4   {       LSnew : = Φ
5           if (Q21.size = 1) then
6              for each small Quarter SQ (with size = 1) in Q43 do
7               { LSnew : = LSnew ∪ Get_Line_Segment(SQ, Q21); }
8           else if (Q43.size = 1 and Q21.size > 2) then
9              for each small Quarter SQ (with size = 2) in Q21 do
10              { LSnew := LSnew ∪ Get_Line_Segment(Q43, SQ); }
11          else if (Q43.size = Q21.size) then
12             LSnew := Get_Line_Segment(Q43, Q21);
13          else if (Q43.size > Q21.size)  then
14             for each small Quarter SQ (with size = Q21.size) in Q43 do
15              { LSnew := LSnew ∪ Get_Line_Segment(SQ, Q21); }
16           else if (Q43.size < Q21.size) then
17             for each small Quarter SQ (with size = Q43.size) in Q21 do
18              { LSnew := LSnew ∪ Get_Line_Segment(Q43, SQ); } }
19  LS : = LS ∪ LSnew
....
```

```
Function Get_Line_Segment(Q43:Quarter, Q21:Quarter) Line Segments;
20        LS4321.Start : = Tri_Peano_Value(LowerLeft(Q43), LowerLeft(Q21));
21        LS4321.End : = LS4321.Start + Q43.size^2 * Q21.Size^2 -1;
22        Return LS4321
End Function
```

함수 2.3 삼각형 쿼터(Tri_Quarter) 분할을 위한 라인세그먼트 계산 함수

```
Function Get_Line_Segment(Q43:Quarter, Q21:Quarter) Line Segments;
1       LS4321.Start : = Tri_Peano_Value(LowerLeft(Q43), LowerLeft(Q21));
2       if Q43.Is_UpperDiagonal and Q21.Is_UpperDiagonal then
3         LS4321.End = LS4321.Start + (Q43.size^2 + Q43.size) / 2 * (Q21.size^2 + Q21.size) / 2 - 1
4       else if Q43.Is_UpperDiagonal and Not Q21.Is_UpperDiagonal then
5         LS4321.End = LS4321.Start + (Q43.size^2 + Q43.size) / 2 * (Q21.size^2) - 1
6       else if Not Q43.Is_UpperDiagonal and Q21.Is_UpperDiagonal then
7             LS4321.End = LS4321.Start + (Q43.size^2) * (Q21.size^2 + Q21.size) / 2 - 1
8       else if Not Q43.Is_UpperDiagonal and Not Q21.Is_UpperDiagonal then
9             LS4321.End = LS4321.Start + (Q43. size^2) * (Q21.size^2) - 1
10      Return LS4321
End Function
```

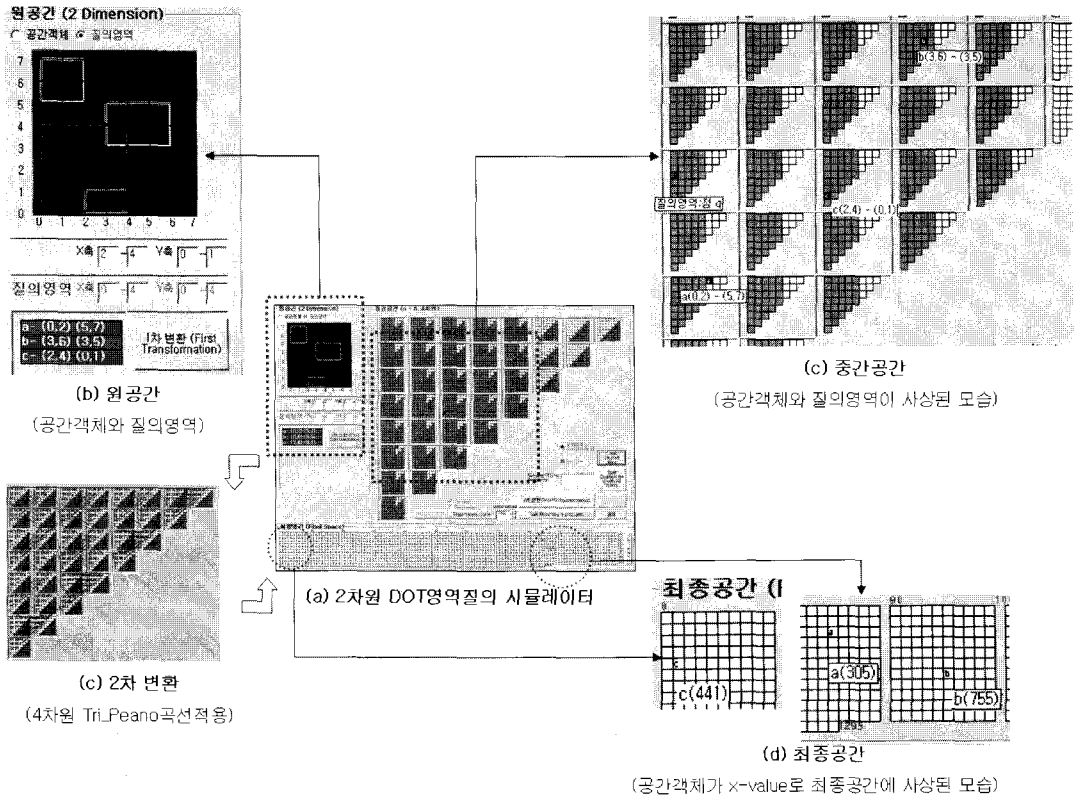
4.5 삼각형 쿼터(Tri-Quarter) 분할을 통한 추가 성능 개선

원 공간을 2차원으로 확장한 경우에도 삼각형 쿼터 분할 방법을 적용할 수 있다. DOT 공간 색인 기법의 특징은 공간 객체를 표현하는 모든 차원의 파라미터가 [객체의 시작점 ≤ 끝점] 인 관계가 성립하므로 모든 공간 객체와 질의 영역은 대각선 위쪽에만 사상되며, 각 차원에 해당하는 사각형의 대각선을 포함하는 위쪽 삼각형만이 쿼터 분할의 대상이 된다. 공간 순서화 곡선 역시 대각선 위쪽만을 운행하는 tri-Peano, tri-Hilbert 곡선을 적용하면 대각선에 걸쳐지는 쿼터가 삼각형 쿼터인 경우에도 질의 영역을 벗어나지 않는다면 연속 운행하게 되므로 대각선에 걸쳐져 작은 크기로 분할되는 쿼터들의 연산을 절감할 수 있다. 이를 위하여 분할된 쿼터가 대각선에 걸쳐지는 경우 삼각형 쿼터(Tri-Quarter)임을 표시하고, 질의 영역에 속하는지를 판별하여 (Is_In_Region함수도 개선필요) 더 이상의 작은 쿼터로 분할하지 않고 삼각형의 면적으로 라인 세그먼트를 계산하여 공간 변환 연산 알고리즘의 성능을 대폭 개선할 수 있다.

삼각형 쿼터(Tri-Quarter) 분할을 통해 라인 세그먼트를 계산하는 방법은 각각 4-3평면과 2-1평면에서 분할된 쿼터가 대각선 위쪽 삼각형인지를 판별하여 다음의 [함수 2.3]의 방식으로 계산한다. 즉 [함수 2.2]의 함수 Get_Line_Segment를 [함수 2.3]과 같이 변경한다.

5. 시뮬레이터를 이용한 성능 분석

1차원 및 2차원 공간 객체에 대하여 DOT 공간 색인 기법을 이용한 영역 질의 과정을 가시적으로 보이며, 이를 이용한 성능 평가를 수행하기 위하여 시뮬레이터를 작성하였다. 시뮬레이터에 의한 2차원 공간 객체에 대한 영역 질의 과정을 예를 들어 그림 15에 보인다. 그림 15의 (a)는 2차원 DOT 시뮬레이터의 전체 화면을 나타내며, 질의 처리를 위한 원공간, 중간 공간, 최종 공간으로 구성되어 있다. 그림 15-(b)는 원 공간의 모습을 표현하고 있으며, 3 개의 공간 객체(a = (0,2,5,7), b = (3,6,3,5), c = (2,4,0,1))와 질의 영역(q = (0,4,0,4))이 주어진 경우를 나타낸다. 이 공간 객체들과 질의 영역을 1차 변환하여 4차원 중간 공간인 해석 모델에 사상한 모습이 그림 15-(c)에 보인다. 이 때 질의영역은 중간 공간 해석

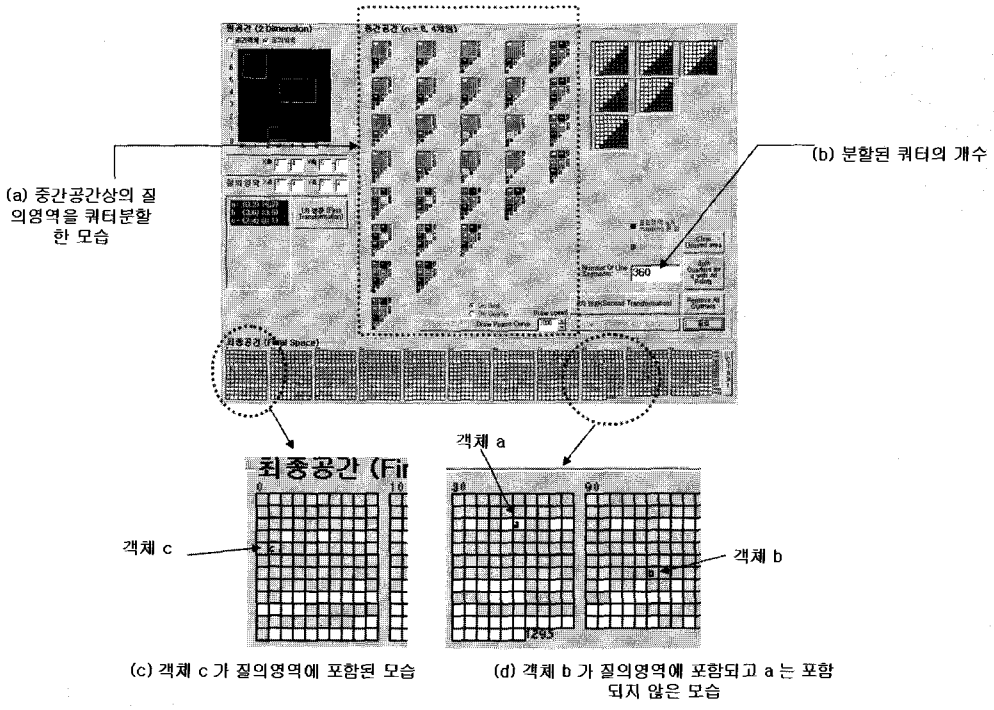


<그림 15> 2차원 공간 객체와 질의 영역이 1차 변환된 모습과 최종 공간에 매핑된 공간 객체
(2차원 DOT 영역 질의 시뮬레이터)

모델에서 대각선 위쪽 삼각형 중 음영처리 되어 표현되어 있다. 그림 15-(c)에 의하여 각 공간 객체들이 각각 하나의 점으로 중간 공간에 표현되어 있으며, 질의 영역에 속하는 2-1 평면은 모두 동일한 영역으로 질의 영역이 설정되었음을 볼 수 있다.

다음의 그림 15-(c)는 4차원 중간 공간 해석 모델에 공간 순서화 곡선으로 4차원 tri_Peano 곡선을 적용하는 과정을 보인다. 이 과정에 의하여 공간 객체가 a = 305, b = 755, c = 441의 x 값을 가지고 최종 공간에 사상된다(그림 15-(d)). 그림 15-(d)는 시뮬레이터 화면의 최종 공간의 모습을 확대한 것으로 격자 모양으로 표현되어 배열 모습을 하고 있으나 0부터 1295까지의 범위를 직선으로 표현한 것으로써 위치를 쉽게 판별하기 위하여 x값을 10개씩 모아놓은 그림이다.

다음의 그림 16는 그림 15의 예를 이용하여 중간 공간에 설정된 질의 영역을 쿼터 분할하여 최종 공간의 라인 세그먼트를 산출하는 과정을 나타내는 예이다. 그림 16-(a)는 질의 영역이 쿼터 분할된 모습을 나타내며, 그림 16-(b)는 이 때 분할된 쿼터의 수를 나타내어 질의 영역이 360개의 쿼터로 분할되었음을 나타낸다. 이렇게 분할된 쿼터가 나타내는 질의 범위(라인 세그먼트)들은 최종 공간 상에 사상되며, 공간 객

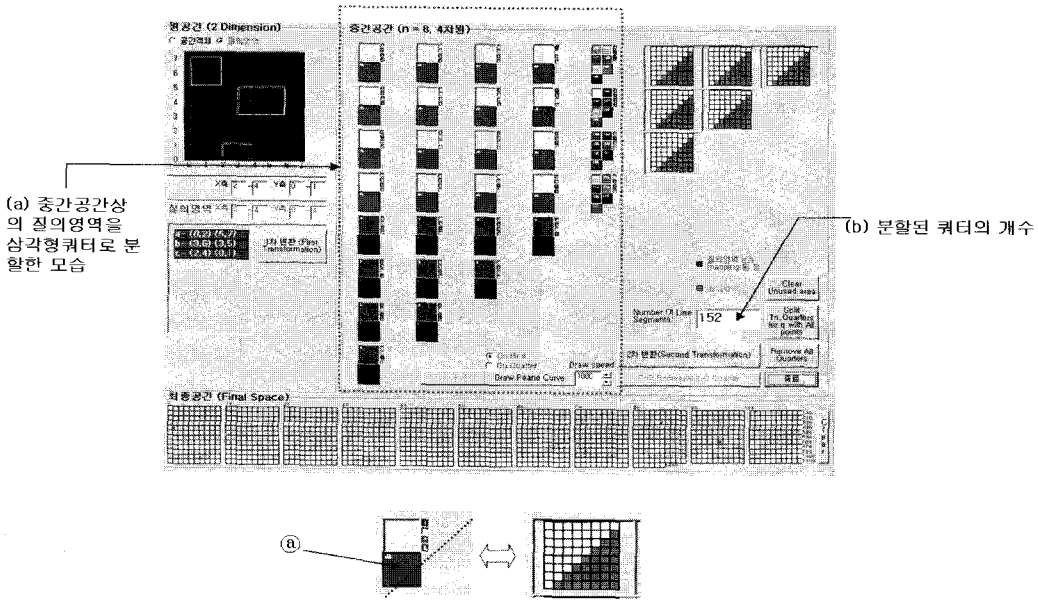


<그림 16> 질의 영역을 쿼터 분할하여 최종 공간에 매핑하는 과정을 나타낸 예

체가 사상된 점 부근을 확대한 그림을 그림 16-(c)와 그림 16-(d)에 보인다. 이 결과, 공간 객체 b, c는 질의 영역에 포함되어 있으며, 공간 객체 a는 질의 영역에 포함되어 있지 않음을 확인할 수 있다.

다음의 그림 17은 그림 15의 예제를 이용하여 중간 공간에 설정된 질의 영역을 삼각형 쿼터(Tri_Quarter)로 분할하는 과정을 보인다. 그림 17-(a)는 질의 영역을 삼각형 쿼터 분할한 결과를 나타내며, 시뮬레이터는 삼각형 쿼터를 그림 17-(c)와 같은 방법으로 표현하고 있다. 또한 그림 17-(b)는 삼각형 쿼터 분할 방법으로 분할된 쿼터의 개수를 나타내며, 152개로 쿼터분할의 경우보다 208회나 공간 변환 연산이 절감되었음을 알 수 있다.

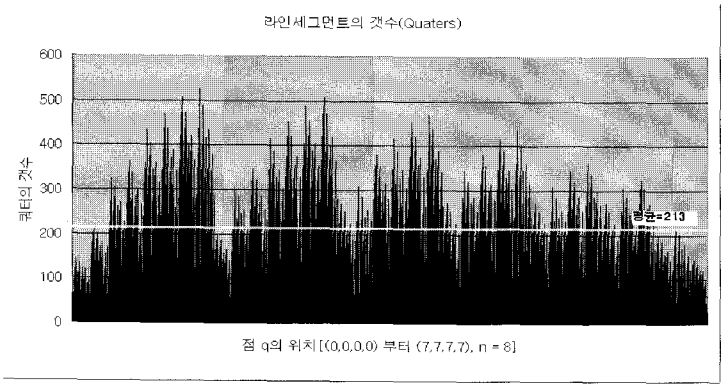
제안된 쿼터 분할 기법의 성능을 분석하기 위하여 질의 영역의 변화에 따른 공간 변환 연산의 횟수를 시뮬레이터를 이용하여 비교하였다. n = 8의 경우, 4차원 중간 공간 상의 모든 점을 질의 영역이 사상된 점 q로 가정하고, 각각 질의 영역이 몇 개의 쿼터로 분할되는지를 조사하였다. 그림 18의 그래프는 점 q의 위치에 따라 분할되는 쿼터의 수를 나타내고 있다. Y축이 분할되는 쿼터의 개수이며 X축은 중간 공간상의 각 점의 위치(0,0,0,0 부터 7,7,7,7)를 나타낸다. 조사결과 분할되는 최대 쿼터의 개수는 525개, 최소 분할되는 쿼터의 개수는 40개, 평균 분할되는 쿼터의 개수는 213개로 평균 변환 연산 횟수는 기대 성능인 $O(4n^2-4n+1)=225$에 근접하지만 평균을 초과하는 빈도가 542회로 전체(1296회)의 41.8%나 되었으며 최대 n^3까지 성능이 악화될 수 있음을 알 수 있다. 이러한 현상은 차원이 확장되면서 발생하는 부가적인 오버헤



<그림 17> 질의 영역을 삼각형쿼터로 분할하는 과정을 나타낸 예

드로써 분할되는 4-3평면의 크기와 2-1평면의 크기를 비교하여 연속된 라인 세그먼트를 찾기 위한 추가적인 쿼터 분할을 원인으로 볼 수 있다.

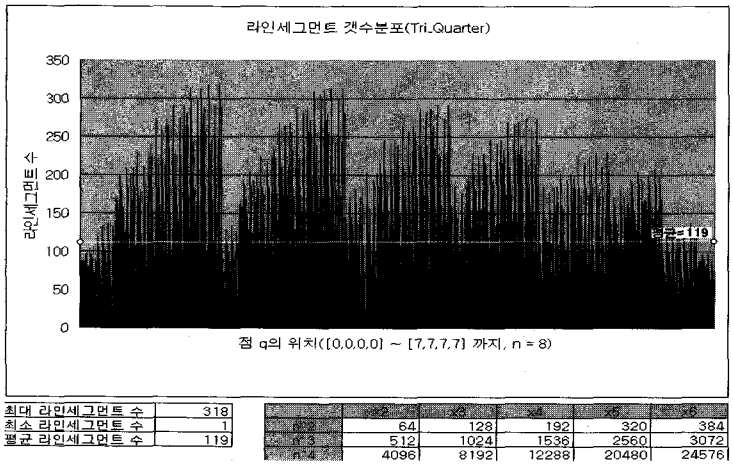
다음의 그림 19는 n = 8인 4차원 중간공간의 모든 점을 질의 영역이 사상된 점 q로 가정하고 2차원 DOT 시뮬레이터를 이용하여 삼각형 쿼터(Tri-Quarter) 분할을 적용한 결과 각각 몇 개의 쿼터로 분할되는지를 조사한 결과이다. 그래프의 X축과 Y축은 그



최대 라인세그먼트 수	525
최소 라인세그먼트 수	40
평균 라인세그먼트 수	213

	x2	x3	x4	x5	x6
n^2	64	128	192	320	384
n^3	512	1024	1536	2560	3072
n^4	4096	8192	12288	20480	24576

<그림 18> 점 q의 위치에 따른 쿼터 분할 횟수



최대 라인세그먼트 수	318
최소 라인세그먼트 수	1
평균 라인세그먼트 수	119

	x2	x3	x4	x5	x6
n^2	64	128	192	320	384
n^3	512	1024	1536	2560	3072
n^4	4096	8192	12288	20480	24576

〈그림 19〉 점 q의 위치에 따른 삼각형 쿼터 분할 횟수

림 18의 경우와 같다. 여기서 분할되는 최대 쿼터의 개수는 318개, 최소 쿼터의 수는 1(질의 영역이 전체인 경우), 평균 분할 쿼터의 개수는 119개로 기대성능(225개)의 52.8%까지로 상당히 개선되어 변환 연산의 횟수가 $O(2n^2)$에 근접하고 있다고 판단할 수 있으며, 또한 기대 성능을 초과하는 빈도가 121회로 전체(1296회)의 9%이므로, 결론적으로 기대 성능을 만족하고 있다고 간주할 수 있다.

6. 결 론

본 논문에서는 다차원 공간 객체를 위한 효율적인 영역 질의 방식을 제안하였다. 영역 질의는 다차원 공간 상에서 질의 영역과 교차 또는 포함되는 객체들을 검색하는 가장 기본적인 공간 연산이다. 영역 질의 처리를 위한 인덱스 기법으로서 공간 순서화 곡선을 이용하여 다차원 공간 객체의 MBR 정보를 1차원 값으로 변환하여 저장하는 DOT(DOuble Transformation) 인덱스 기법이 알려져 있다.

본 연구에서는 DOT 색인 방식을 k차원 공간 객체에 적용하기 위하여 2k차원의 중간 공간이 2차원 배열의 재귀적인 연속 공간임을 밝히고, 공간 객체의 표현 방식, 다차원 공간 상의 공간 순서화 곡선의 운행 방식, 영역 질의 방법을 제안하였다. 제안된 기법에서는 영역 질의 알고리즘의 성능을 보장하기 위하여 중간 공간을 공간 순서화 곡선이 연속되는 최대 크기의 쿼터로 분할하여 공간 변환 연산의 비용을 절감하였다. 이때 질의 영역을 최종 공간의 라인 세그먼트로 변환하는 알고리즘의 복잡도가 $O((2n-1)^k)$이며, 대각선 위쪽 삼각형만을 분할하여(Tri-Quarter 분할) 성능을 개선할 수 있음을 보였다. 또한, 제안된 기법에 의한 다차원 영역 질의 처리 과정을 시각적으로 확인할 수 있는 시뮬레이터를 구현하였으며, 이를 이용한 성능 평가 결과를 보였다.

실용화를 위한 몇 가지 연구 과제가 남아 있으며, 기존의 다차원 인덱싱 방식을 이용한 영역 질의 방식과의 비교 연구를 진행할 예정이다.

참고 문헌

1. A. Guttman, "R-trees: A Dynamic Index Structures for Spatial Searching," *Proc. ACM SIGMOD*, 1984, pp. 47-57.
2. N. Beckmann, H. Kriegel, and R. Schneider, "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD*, 1990, pp. 322-331.
3. T. Brinkhoff, H. P. Kriegel, and B. Seeger, "Efficient Processing of Spatial Joins Using R-trees," *Proc. ACM SIGMOD*, 1993, pp. 237-246.
4. K. Hinrichs, J. Nievergelt, "The Grid File : A Data Structure Designed to Support Proximity Queries on Spatial Objects," *Proc. Workshop on Graph Theoretic Concepts in Computer Science*, 1983, pp. 100-113.
5. J. Song, K. Whang, Y. Lee, M. Lee, and S. Kim, "Spatial Join Processing Using Corner Transformation," *IEEE Trans. Knowledge and Data Engineering*, vol. 11, no. 4, 1999, pp. 688-695.
6. C. Faloutsos, and S. Roseman, "Fractals for Secondary Key Retrieval," *Proc. PODS*, 1989, pp. 247-252.
7. C. Faloutsos, and Y. Rong, "DOT : A Spatial Access Method Using Fractals," *Proc. 7th Int'l. Conf. on Data Engineering*, 1991, pp. 152-159.
8. 백현, 유용혁, 윤지희, 이건배, "DOT 공간색인 기법을 이용한 효율적인 공간조인 처리 기법," *Korean DataBase Conference 2000 학술발표 논문집*, 2000, pp. 65-74.
9. C. Faloutsos, and T. Sellis, and N.Roussopoulos, "Analysis of Object Oriented Spatial Access Methods," *Proc. ACM SIGMOD*, 1987, pp. 426-439.
10. H. Kriegel, M. Schiwietz, R. Schneider, and B.Seeger, "Performance comparison of Point and Spatial Access Methods," *SSD, Lecture Note*, 1989, pp. 89-114.
11. J. Orenstein, F. Manola, "PROBE Spatial Data Modeling and Query Processing in an Image Database Applications," *IEEE Trans. on Software Engineering*, Vol 14, No. 5, 1988.
12. J. T. Robinson, "The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes," *ACM SIGMOD Conf.*, 1981, pp. 10-18.
13. A. Henrich, H.-W. Six, and P. Widmayer, "The LSD Tree: Spatial Access to Multidimensional Point and Non Point Objects," *Proc. 15th VLDB Conf.*, 1989, pp 45-53.
14. C. Faloutsos, and Y. Rong, "Spatial Access Methods Using Fractals : Algorithms and Performance Evaluation," *UMIACS-TR-89-31, CS-TR-2214, Univ. of Maryland*, 1989, pp. 1-17.
15. O. Gunther, "The Cell Tree: An Index for Geometric Data," *Memorandum No. UCB/ERL M86/89, Univ. of California, Berkeley*, 1986.
16. J. Lawder and P. King, "Querying Multi-dimensional Data Indexed Using the Hilbert Space-Filling Curve," *ACM SIGMOD Record*, Vol. 30, No. 1, 2001, pp. 19-24.
17. B. Moon, H. V. Jagadish, C. Faloutsos, and J. Saltz, "Analysis of the Clustering Properties of Hilbert Space-Filling Curve," *IEEE Trans. on Knowledge and*

Data Engineering, Vol. 13, No. 1, 2001, pp. 124-141.

18. H. Dai and H. Su, "Approximation and Analytical Studies of Inter-cluster Performances of Space-Filling Curves," *Discrete Mathematics and Theoretical Computer Science*, 2003, pp. 53-68.

백 현
1987년 숭실대학교 전자계산학과(공학사)
1997년 한림대학교 경영대학원(경영학석사)
1999년 한림대학교 컴퓨터공학과
2001년~현재 (주)시스게이트 기술연구소
관심분야 : 공간데이터베이스, 데이터 마이닝, IT서비스관리

윤지희
1982년 2월 한양대학교 전자공학과 졸업(학사)
1985년 3월 일본 구주대학교 정보공학과 졸업(석사)
1988년 3월 일본 구주대학교 정보공학과 졸업(박사)
1998년 3월~1999년 2월: 미국 UCLA대학교 전산학과 방문교수
1988년 4월~현재 한림대학교 정보통신공학부 교수
관심분야 : 바이오인포매틱스, 데이터 마이닝, XML, 공간 데이터베이스/GIS

원정임
1992년 2월 한림대학교 전자계산학과 졸업(학사)
1997년 8월 한림대학교 컴퓨터공학과 졸업(석사)
2004년 2월 한림대학교 컴퓨터공학과 졸업(박사)
2000년 3월~2004년 2월 한림대학교 교양교육부 강의전담교수
2004년 3월~2006년 2월 연세대학교 컴퓨터과학과 연구교수
2006년 3월~2006년 6월 서울대학교 유전자이식연구소 선임연구원
2006년 7월~현재 한양대학교 정보통신대학 정보통신학부 연구교수
관심분야 : 데이터베이스 시스템, 데이터 마이닝, XML 응용, 바이오 정보공학, 데이터베이스 보안, 이동객체 데이터베이스, 텔레매틱스