# 대용량 센서 데이터 아카이빙을 위한 색인 분할 기법†

## An Index Splitting Technique for Numerous Sensor Data Archiving

조대수* / Dae-Soo Cho

**요 약**

센서 데이터는 대용량이며 지속적으로 발생하는 특징이 있다. 따라서 대용량의 센서 데이터로부터 특정 데이터를 효과적으로 검색하기 위해서는 색인의 개발이 요구된다. 센서 데이터 색인은 데이터 아카이빙을 지원하기 위해서 일정한 시간이 지난 과거의 데이터를 효과적으로 삭제할 수 있어야 한다. 본 논문에서는 대용량 센서 데이터 아카이빙을 지원하기 위해 색인 분할 기법을 제안하고 구현하였다. 분할된 각각의 색인들은 가상색인으로 구성되어, 외부에서는 하나의 색인으로 보인다. 실험 결과 색인의 생성비용은 총 100,000번의 삽입 연산에 대해서 최대 8%의 성능 향상을 보였으며, 삽입되는 데이터의 개수가 많아질수록 성능이 더 향상됨을 보였다. 영역질의의 경우 각 질의영역이 적을수록, 시간도메인의 크기가 커질수록 큰 성능 향상을 보였다.

**Abstract**

Sensor data have the characteristics such as numerous and continuous data. Therefore, it is required to develop an index which could retrieve a specific sensor data efficiently from numerous sensed data. The index should have an efficient delete operation for the past data to support the data archiving. In this paper, we have proposed and implemented an index splitting technique to support the sensor data archiving. These splitted indexes compose of a virtual index (that is, index management component), which is shown as single tree from outside. Experimental results show that in the case of 100,000 insert operations the splitted index performs 8% better than the traditional TB-tree maximumly. And the splitted index outperforms TB-tree with retrieving queries when the region of query is small and the size of time domain is large.

**주요어 :** 센서데이터, 아카이빙, 색인, 이동체, 분할기법

**Keyword :** Sensor Data, Archiving, Index, Moving Object, Splitting Technique

† 이 논문은 2005년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음 (KRF-2005-003-D00285)

■ 논문접수 : 2007.2.6 ■ 심사완료 : 2007.4.25

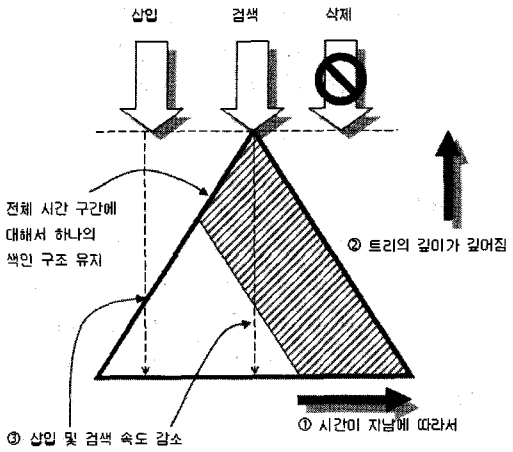
* 교신저자 동서대학교 컴퓨터정보공학부 조교수 (dscho@dongseo.ac.kr)

## 1. 서론

최근 새로운 IT의 패러다임인 유비쿼터스 컴퓨팅이 다양한 분야(특히, IT 분야)에서 관심이 집중되어 있다. 유비쿼터스 컴퓨팅에서는 (1)다수의 센서가 사물에 부착되어 있으며, (2)이들 센서로부터 수많은 센서 데이터들이 발생하며, (3)수집되는 센서 데이터로부터 상황인지(Context Awareness)를 통해 사용자 및 시스템이 원하는 서비스가 제공될 수 있다. 유비쿼터스 컴퓨팅에서 상황인지는 5W1H를 포함하고 있으며, 센싱을 통해 상황을 인지할 수 있다. 5W는 Where, Who, When, What, Why를 의미하고, 1H는 How를 의미한다[1]. 본 논문은 5W1H 중에서 위치인식(Location-aware)을 목적으로 Where(즉, 위치정보)에 대한 센싱에서 요구되는 색인을 대상으로 한다. 위치정보 센싱에 사용되는 센서로는 RFID, IR, GPS, Ultrasound, Visual Tag 등이 있다.

시간에 따라 자신의 위치가 지속적으로 변하는 객체를 이동체(Moving Objects)라 하며, 이동체에 대해서 GPS와 같은 센서에 의해 수집되는 다수의 위치정보를 다루는 색인을 이동체 색인이라 한다. 현재 대부분의 색인 연구에서는 효과적인 저장, 검색 연산을 지원하는 것을 목표로 하고 있으며, 데이터 아카이빙에 대해서는 크게 다루어지고 있지 않다. 데이터 아카이빙에 대한 필요성을 예를 들어 설명하면 다음과 같다. 1백만 개의 이동체에 대해서 매 5분마다 위치정보를 센싱해야 할 경우에, 하나의 이동체마다 매일 288개의 센서 데이터가 발생한다. 따라서 1백만 개 이동체에 대해서는 2억 8천 8백만 개의 센서 데이터가 매일 발생하므로, 대용량 데이터에 대한 효과적인 저장, 검색 기능뿐 아니라, 반드시 과거 데이터에 대한 아카이빙 기능이 요구된다.

기존의 색인 방식은 색인의 생성 시점부터 현재까지의 모든 데이터를 하나의 색인으로 관리하는 전체 색인 방식이다. 따라서 <그림 1>과 같이 ① 시간이 지남에 따라서, ②트리의 높이가 증가하므로 ③삽입 및 검색 연산에 대해서 속도가 감소되는 단점을 가지고 있다. 또한 효과적인 삭제 연산을 지원할 수 없기 때문에, 지속적으로 유입되는 센서 데이터에 대해서 물리적으로 하나의 색인으로 계속 관리할 수 없는 상황이 발생하게 된다. 오래된 위치정보에 대한 삭제 연산은 다음과 같은 경우에 발생한다. 첫째, 시간이 지남에 따라서 색인의 크기가 무한히 증가하는 것을 방지하기 위해서 가장 과거의 위치정보를 삭제할 수 있다. 둘째, 실제 응용에서 사용되는 위치정보는 저장되어야 할 유효 기간을 갖는데, 유효 기간이 지난 위치정보는 시스템의 성능을 위해서 삭제하거나 백업될 수 있다.




<그림 1> 전체 색인 방식의 문제점

본 논문에서는 전체 색인 방식의 문제점을 해결하기 위해서 (1)대용량 센서 데이터(위치 데이터)를 효과적으로 관리하고, (2)과거의 위치 데이터에 대한 아카이빙을 지원하기 위해서 색인 분할 기법을 제안하였다. 즉, 새로운 색인 자체를 제안하는 것이 아니라, 기존에 개발된 색인을 시간 도메인에 대해서 여러 개의 색인으로 분할하여 관리하는 방법을 제안하였다. 본 논문에서 제안하는 색인 기법을 TB-tree에 적용한 결과, (1)영역 질의, (2)타임 슬라이스 질의, (3)복합 질의 각각에 대해서 유사하거나, 우수한 성능을 보였다.

이 논문의 구성은 다음과 같다. 2장에서는 이동체 색인에 대한 관련연구를 살펴보고, 3장에서는

데이터 아카이빙을 지원하기 위한 색인 분할 기법을 제안한다. 4장에서는 제안한 색인 분할 기법을 TB-tree에 대해서 적용하고, 성능 평가한 결과를 기술한다. 그리고 5장에서 결론과 향후 연구내용을 기술한다.

## 2. 관련연구

위치데이터에 대한 색인은 이동체에 부착된 센서로부터 수집되는 좌표 정보(x, y)와 센싱이 발생한 시간 정보(t)로 구성된 3차원 시공간을 다루고 있다. 이동체 색인은 각 타임스탬프마다 2차원의 공간 색인을 생성함으로써, 시간 도메인을 공간 도메인과 별도로 취급하는 색인 방법과 시공간 도메인을 동시에 3차원으로 처리하는 색인 방법으로 나눌 수 있다[2].

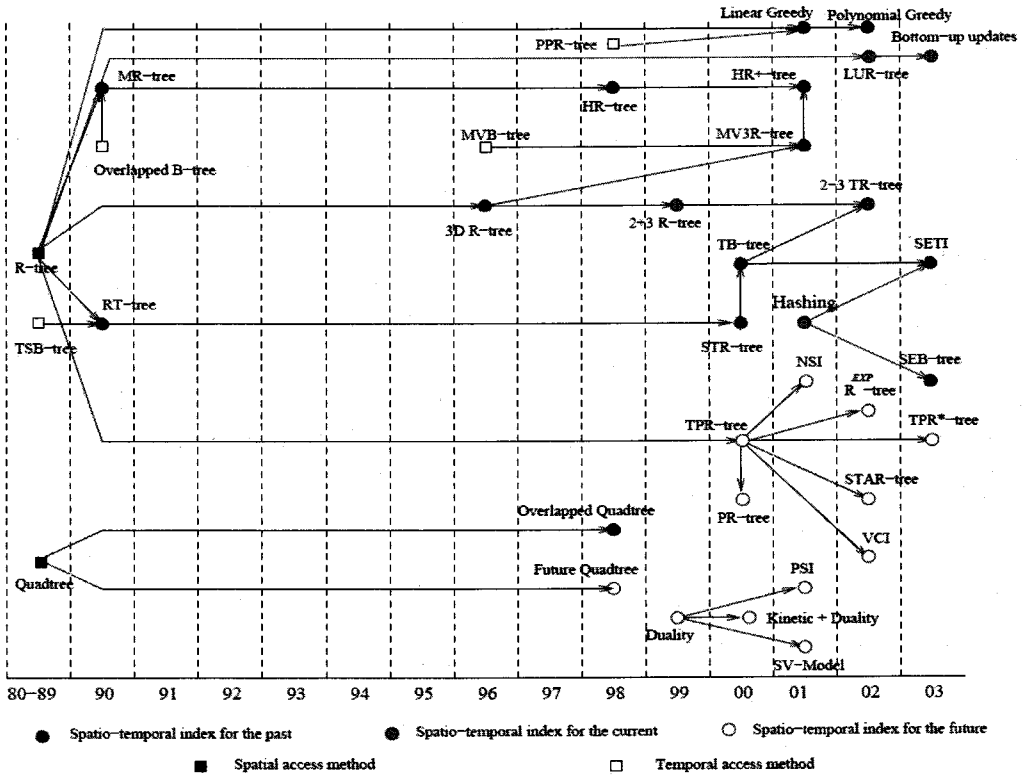
시간 도메인과 공간 도메인을 별도로 취급하여, 각 타임스탬프마다 2차원의 공간 색인을 사용하는 이동체 색인에는 대표적으로 HR-tree[3]가 있다. HR-tree는 각 타임스탬프 마다 2차원 R-tree가 존재하며, 연속되는 두 타임 스탬프간에 변화가 있는 경우에는 새로운 노드를 생성하고, 그렇지 않는 경우에는 과거의 노드를 공유한다. 따라서 HR-tree는 시간의 흐름에 따라 객체의 변화가 적은 경우에는 적합하나 객체의 위치 변화가 빈번한 경우 색인의 크기가 커지는 문제점이 있다.

시간 도메인과 공간 도메인을 3차원 시공간 도메인으로 처리하며, 영역 질의와 궤적 질의를 동시에 고려한 이동체 색인에는 STR-tree 및 TB-tree[4], OP-tree[5] 등이 있다. STR-tree는 R-tree 기반이며 궤적을 처리할 수 있도록 기존 R-tree[6]를 확장하였으며, 공간 근접성(Spatial Closeness) 뿐만 아니라 궤적 보존(Trajectory Preservation)을 시도하는 색인 구조이다. STR-tree의 보존 파라미터(Preservation Parameter) p를 사용하여 공간적인 특성과 궤적 보존성을 조절할 수 있으며, 보존 파라미터가 작을수록 궤적 보존성은 낮아지고 공간 근접성은 높아지는 특성이 있다. TB-tree는 STR-tree에 비해 궤적 질의의 성능을 개선하기 위하여 단말 노드에 동일한 궤적만을 저장하고 있다. OP-tree는 객체 단순화 방법으로 기존 MBB (Minimum Bounding Rectangle) 대신 MBOP (Minimum Bounding Octagon Prism)을 사용하여, 더 세밀하게 객체를 단순화시킴으로써 사장 영역이 줄어드는 장점이 있으나, 노드의 팬 아웃이 낮아지는 단점이 있다.

3차원 시공간 데이터에 대한 효과적인 색인 방법은 질의 유형에 매우 의존적이며, 이러한 질의 유형은 (1)영역질의, (2)타임 슬라이스 질의, (3)복합 질의 등으로 구분될 수 있다. 영역질의는 특정 시간 간격에서의 이동체 위치에 대한 질의로서, 3DR-tree[7], MV3R-tree[8] 등이 우수한 성능을 보인다. 타임 슬라이스 질의는 특정 시점에서의 이동체 위치에 대한 질의로써, HR-tree, HR+-tree 등이 우수한 성능을 보인다. 복합 질의 및 궤적 질의는 이동체의 궤적(특정 이동체의 시간에 따라 변경된 위치정보)에 대한 질의로써, STR-tree, TB-tree 등이 우수한 성능을 보인다.

분류하는 방법에 따라서 시공간 색인은 <그림 2>와 같이 공간 중심의 Quad-tree 계열의 색인 방법과 데이터 중심의 R-tree 계열의 색인 방법으로 분류할 수 있으며, 현재 데이터에 대한 색인 방법, 과거 데이터에 대한 색인 방법, 미래 데이터에 대한 색인 방법으로 분류할 수 있다.

기존의 이동체 색인에 대한 연구에서는 대부분 R-tree 계열의 데이터-중심 색인을 확장하여 사용하고 있으며, 각각의 질의 유형에 효과적인 색인 방법들을 제안하고 있다. 그러나, R-tree 계열의 색인들은 모두 R-tree의 특징인 균형이 잡힌 균형 색인 구조이므로, 데이터 아카이빙을 위해서는 전체 색인 구조를 재구성해야 하는 문제점이 있으므로, 현재까지의 색인 구조로는 데이터 아카이빙을 효율적으로 지원하기 어렵다. 따라서 대용량이며, 지속적인 특성을 갖는 센서 데이터의 아카이빙을 지원하기 위해서는 새로운 색인 구조가 필요하다.

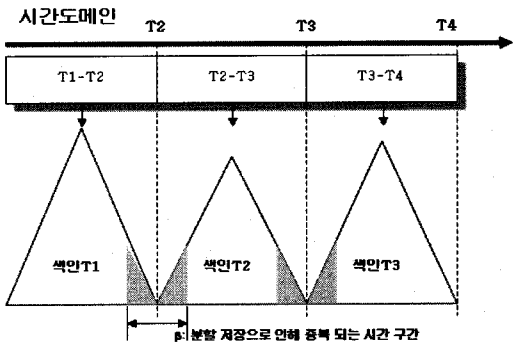



<그림 2> 시공간 접근 메소드 분류[9]

## 3. 아카이빙을 위한 색인분할 기법

일반적으로 색인은 모든 단말 노드까지의 깊이가 동일한 균형이 잡힌 균형 트리와 단말 노드의 깊이가 서로 다른 비균형 트리로 구분될 수 있다. 균형 트리는 항상 트리의 균형을 맞추기 위해서 데이터 삭제 시 트리 구조에 대한 재구성 오버헤드가 비균형 트리보다 큰 단점이 있는 반면, 모든 단말 노드에 대해서 깊이가 동일하기 때문에, 데이터 검색 시 일정한 수준의 응답시간을 보장하는 장점이 있다.

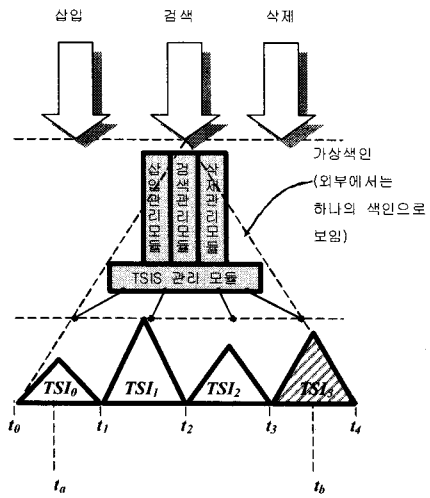
대용량 위치 데이터에 대한 대부분의 색인 구조에서는 검색의 효율성을 높이기 위해서 균형 트리인 R-tree를 기반으로 색인을 설계하고 있다. 따라서 본 논문에서는 과거 데이터에 대한 아카이빙을 지원하기 위해서 시간 도메인에 대해서는 비균형 트리를 구성하여 시간을 분할하고, 각각 분할된 시간 영역에 대해서 균형 트리를 구성하는 하이브리드 형태의 색인을 제안한다. 시간 도메인에 대해서는 비균형 트리를 구성하는 이유는 재구성 오버헤드가 없기 때문에, 과거 데이터의 아카이빙을 효율적으로 처리할 수 있기 때문이다.




<그림 3> 시간도메인 비중첩 분할 색인

<그림 3>은 시간 도메인에 대한 시간 분할시 인접한 노드 간에 시간의 중첩이 발생하지 않는 구조로 색인이 관리되고 있음을 보여주고 있다. 위치 정보는 궤적(trajectory) 추출을 위해 (이전위치, 현재위치)와 같은 형태로 저장되므로, 이전 위치를 센싱한 시간과 현재 위치를 센싱한 시간사이의 간격이 색인의 시간 도메인 분할 시간에 걸쳐지는 경우가 발생한다. 따라서 시간도메인 비중첩 분할 색인에서는 인접한 두 개의 분할 색인에서 각각 동일한 위치정보(이전위치, 현재위치)를 중복해서 저장해야 한다.

이 장에서는 본 논문을 통해 구현된 색인 시스템에 대해서 시스템 구조 및 삽입, 삭제, 검색 알고리즘에 대해서 살펴본다. <그림 4>는 본 논문에서 구현한 색인 시스템의 구조를 보이고 있다. 본 논문에서는 전체 색인을 시간 도메인으로 분할해서 시간 세그먼트 색인(TSI, Time-Segmented Index)으로 나누어서 관리하며, 외부에서는 하나의 색인 구조로 보이도록 가상색인으로 구현된다. TSIS 관리 모듈은 시간 세그먼트 색인 집합 TSIS = {$TSI_1$, $TSI_2$, $TSI_3$, ..., $TSI_n$}를 관리한다. 즉, 시간 세그먼트 색인 집합 TSIS(TSI Set)은 시간 구간 [tx,ty)별로 쪼개어진 시간 세그먼트 색인 $TSI_x$의 집합을 의미한다.



<그림 4> 시간 분할 색인 관리 모듈의 개념

TSIS 관리 모듈은 몇 가지 주요 함수를 포함하고 있다. 첫째, 주어진 시간 t(단, $t_x \le t < t_{x+1}$, $[t_x,t_y)$는 특정 시간 세그먼트 색인에 대한 시간 도메인임)의 데이터를 관리하는 시간 세그먼트 색인을 넘겨주는 매핑 함수 $f_{tsi}(t)$가 있다. 즉, $f_{tsi}(t) = TSI_x$. 둘째, 주어진 시간 간격 $[t_a, t_b)$에 대해서 겹치는 구간의 시간 세그먼트 집합 $TSI_{[ta,tb)}$을 넘겨주는 함수 $f_{tsis}(t_a, t_b)$가 있다. 즉, $f_{tsis}(t_a, t_b) = TSI_{[ta,tb)} = \{ f_{tsi}(t_a)=TSI_i, f_{tsi}(t_b)=TSI_k \} \cup \{ TSI_i | TSI_i \subset TSIS, t_i<t_j<t_k \}$.

<그림 4>에 대해서 위 두 함수의 예들 들면 다음과 같다.

- ▶ $f_{tsi}(t_a)=TSI_0$ : 시간 $t_a$의 데이터를 관리하는 시간 세그먼트 색인
- ▶ $f_{tsi}(t_b)=TSI_3$ : 시간 $t_b$의 데이터를 관리하는 시간 세그먼트 색인
- ▶ $f_{tsis}(t_a, t_b) = TSI_{[ta,tb)}$ : 시간 간격 $[t_a, t_b)$에 대해서 겹치는 구간의 시간 세그먼트 집합
  $= \{ TSI_0, TSI_3 \} \cup \{ TSI_i | TSI_i \subset TSIS, t_0<t_j<t_3 \}$
  $= \{ TSI_0, TSI_3 \} \cup \{ TSI_1, TSI_2 \}$
  $= \{ TSI_0, TSI_3 \} \cup \{ TSI_1, TSI_2 \}$
  $= \{ TSI_0, TSI_1, TSI_2, TSI_3 \}$

TSIS 관리 모듈에 의한 데이터의 삽입, 영역질의, 궤적질의, 복합질의 알고리즘은 (알고리즘 1)~(알고리즘 4)와 같다. (알고리즘 1)은 주어진 이동체에 대해서 (이전위치, 현재위치)의 정보가 하나의 선객체로 삽입될 경우에 처리 과정을 보이고 있다. TSIS 모듈에 의해 가장 최근에 생성된 분할 색인인 $TSI_k$가 해당 선객체의 시간 범위를 완전히 포함할 경우에는 $TSI_k$에 데이터의 삽입을 요청한다. 그렇지 않은 경우에는 새로운 분할색인 $TSI_{k+1}$을 생성하고, $TSI_k$와 $TSI_{k+1}$에 중복해서 데이터를 삽입한다.

(알고리즘 2)는 주어진 영역에 겹치는 이동체의 ID를 검색하는 과정을 보이고 있다. 먼저, 시간 간격 $[t_a, t_b)$에 대해서 겹치는 구간의 시간 세그먼트 집합을 구하고, 이에 포함된 각각의 분할 색인에 대

| Algorithm TSIS::Insert( MOID, LineSegment$_{[ta,tb)}$ )<br>- MOID: ID of moving object<br>- LineSegment$_{[ta,tb)}$ : line segment object from $t_a$ to $t_b$ | |
|---|---|
| AI1 | Let $TSI_k$ be the current time-segmented index managed by TSIS module; |
| AI2 | If ($t_k \le t_b < t_{x+1}$)<br>$TSI_k$.Insert( MOID, LineSegment$_{[ta,tb)}$ ); |
| AI3 | Else<br>$TSI_k$.Insert( MOID, LineSegment$_{[ta,tb)}$ );<br>CreateNewTSI($t_{k+1}$);<br>$TSI_{k+1}$.Insert( MOID, LineSegment$_{[ta,tb)}$ ); |

(알고리즘 1) TSIS 관리 모듈의 데이터 삽입 알고리즘

| Algorithm RESULT TSIS::RangeSearch( Ranage$_{[ta,tb)}$ )<br>- Range$_{[ta,tb)}$ : some spatial range with time interval from $t_a$ to $t_b$<br>- RESULT : ID list of moving objects | |
|---|---|
| AR1 | $TSI_{[ta,tb)} = f_{tsis}(t_a, t_b)$; |
| AR2 | foreach $TSI_k$ in $TSI_{[ta,tb)}$<br>Add $TSI_i$.RangeSearch( Ranage$_{[ta,tb)}$ ) to RESULT; |
| AR3 | Remove the duplications in RESULT; |
| AR4 | Return RESULT; |

(알고리즘 2) TSIS 관리 모듈의 영역 질의 알고리즘

| Algorithm RESULT TSIS::TrajectorySearch( MOID, Ranage$_{[ta,tb)}$ )<br>- MOID : ID of moving object<br>- Range$_{[ta,tb)}$ : some spatial range with time interval from $t_a$ to $t_b$<br>- RESULT : a trajectory of the moving object with MOID | |
|---|---|
| AR1 | $TSI_{[ta,tb)} = f_{tsis}(t_a, t_b)$; |
| AR2 | foreach $TSI_k$ in $TSI_{[ta,tb)}$<br>Add $TSI_i$.TrajectorySearch( MOID, Ranage$_{[ta,tb)}$ ) to RESULT; |
| AR3 | Remove the duplications in RESULT; |
| AR4 | Make several trajectories in RESULT to be connected as one trajectory; |
| AR5 | Return RESULT; |

(알고리즘 3) TSIS 관리 모듈의 궤적 질의 알고리즘

해서 영역질의를 요청한다. 마지막으로 중복 검색된 데이터에 대해서 중복을 제거하고 그 결과를 리턴한다.

(알고리즘 3)는 주어진 이동체 ID에 대해서 주어진 영역내의 궤적을 검색하는 과정을 보이고 있다. 먼저, 시간 간격 [$t_a$, $t_b$)에 대해서 겹치는 구간의 시간 세그먼트 집합을 구하고, 이에 포함된 각각의 분할 색인에 대해서 궤적질의를 요청한다. 그리고, 색인이 분할되는 시점에 중복 저장된 궤적을 제거하고, 마지막으로 여러 분할 색인으로부터 검색된 궤적들을 하나의 궤적으로 통합하여 리턴한다.

(알고리즘 4)는 영역질의 및 궤적질의를 순차적으로 호출함으로서, 복합질의를 처리하는 과정을 보이고 있다.

| Algorithm RESULT TSIS::CombinedSearch( $InnerRange_{[ta,tb)}$, $OuterRange_{[tx,ty)}$)<br>- $InnerRange_{[ta,tb)}$ : inner range of combined search to search some MOIDs<br>- $OuterRange_{[ta,tb)}$ : outer range of combined search to extract trajectories of the corresponding MOIDs<br>- RESULT : list of trajectories of moving objects | |
|---|---|
| AC1 | $TSI_{[ta,tb)} = f_{tsis}(t_a, t_b)$; |
| AC2 | foreach $TSI_k$ in $TSI_{[ta,tb)}$<br>Add $TSI_k$.RanageSearch( $Ranage_{[ta,tb)}$ ) to MOID list; |
| AC3 | Remove the duplications in MOID; |
| AC4 | foreach ID in MOID<br>Add TSIS.TrajectorySearch( ID, $Ranage_{[ta,tb)}$ ) to RESULT; |
| AC5 | Return RESULT; |

(알고리즘 4) TSIS 관리 모듈의 복합 질의 알고리즘

## 4. 성능평가 결과

이 절에서는 아카이빙을 위해 적용된 색인 분할(Index Splitting) 기법의 성능을 평가한다. 분할 대상 색인으로는 이동체 색인 중에서 시간으로 정렬이 잘 되어있는 TB-tree를 사용한다. 즉, 원래의 TB-tree와 분할된 TB-tree($TB_S$-tree로 표기함)와의 성능을 비교하고자 한다. 각각의 트리는 C#언어로 구현되었으며, 윈도우즈XP 운영체제, 1GB 메인메모리, P-IV 3.06Ghz CPU가 탑재된 PC를 이용하여 실험하였다.

### 4.1 실험 데이터 셋

이동체 생성을 위한 대표적인 도구로 GSTD(Generate Spatio-Temporal Data) 생성기[10]와 Brinkhoff의 네트워크 데이터 생성기[11] 그리고 IBM의 CitySimulator[12]가 있다. 이 논문에서는 GSTD와 네트워크 데이터 생성기를 사용하여 생성된 데이터셋(dataset)에 대하여 실험을 하였다.

GSTD로 생성한 데이터셋은 이동체의 위치 정보를 시공간의 점으로 표현되지만, 성능 평가를 위해서 색인을 생성할 경우에, 이동체의 궤적은 시공간의 선분으로서 표현되어야 한다. 본 논문에서는 GSTD에 의해 생성된 데이터셋에서 동일 궤적 ID를 가지는 점들에 대해서 시간 순서상 연속되는 두 점을 하나의 선분으로 변환하여 색인에 삽입하였다.

GSTD를 사용할 경우, 이동체의 개수(ON), 위치보고 횟수(RN), 이동 시나리오(S) 등의 매개변수에 따라서 여러 가지 유형의 데이터 셋이 생성될 수 있다. 본 논문에서는 ON, RN을 고정하고, S를 변경하는 데이터 셋 유형인 D(2,3,S)와 ON, S를 고정하고 RN를 변경하는 데이터 셋 유형인 D(1,RN,4)를 사용하여 실험한다.

<표 5> 실험 데이터 셋 유형

| 데이터 셋 유 형 | 이동체수 | 보고횟수 | 이동시나리오 [10] | 비 고 |
|---|---|---|---|---|
| D(2,3,S)<br>단, S는 1~6임 | 고정($10^2$) | 고정($10^3$) | 시나리오1 | 저장되는 데이터 개수를 고정하고, 이동 시나리오에 대한 실험을 위해 사용 |
| | | | 시나리오2 | |
| | | | 시나리오3 | |
| | | | 시나리오4 | |
| | | | 시나리오5 | |
| | | | 시나리오6 | |
| D(1,RN,4)<br>단, RN은 2~5임 | 고정($10^1$) | $10^2$ | 시나리오4 | 시간에 따라 지속적으로 데이터가 축적 |
| | | $10^3$ | | |
| | | $10^4$ | | |
| | | $10^5$ | | |

매개변수에서 ON과 RN의 값은 각각 $10^{ON}$개의 이동체 개수, $10^{RN}$개의 위치보고 횟수를 의미한다. 매개변수 S는 시나리오 번호를 의미한다. GSTD 알고리즘은 초기 데이터 분포, 이동방향, 이동속도, 크기변화 등의 매개변수에 값에 따라 다양한 이동 시나리오가 가능하며, [10]에서는 6가지의 예제 시나리오를 제안하고 있다.

### 4.2 실험 질의 셋

성능 평가 실험은 색인의 생성 비용과 검색 비용으로 나누어서 수행하며, 이 절에서는 검색 비용을 측정하기 위한 검색 질의의 유형과 검색 질의 셋에 대하여 기술한다. 검색 질의의 유형은 다음과 같이 구분될 수 있다.

- ▶ 영역질의: 시간과 공간의 각 축에 대해서 일정한 크기를 갖는 영역을 입력으로 하고, 해당 영역 내에 포함된 이동체를 검색하는 질의이다.
- ▶ 타임 슬라이스 질의: 영역질의의 특수한 경우로서, 시간축에 대해서 일정한 크기를 주는 것이 아니라, 특정 시간값을 주고, 공간에 대해서만 일정 영역을 입력으로 주고, 해당 시간 값과 공간 영역에 대해서 이동체를 검색하는 질의이다.
- ▶ 복합질의: 영역질의에 의해 특정 이동체를 선택하고, 선택된 이동체의 궤적에 대해서 궤적선분(궤적의 일부분)을 추출하는 두 단계로 구분된다.
  - ※ 복합 질의의 예 : "2006년 11월 29일 10:00 에서 10:30 까지 부산은행 장전동지점 앞 사거리를 지난 차량들의 다음 한 시간까지의 궤적을 추출하라"
    - "2006년 11월 29일 10:00에서 10:30까지 부산은행 장전동지점 앞 사거리를 지난 차량들" : 영역 질의를 통하여 이동체(궤적)를 선택하는 과정
    - "다음 한 시간까지 궤적을 추출하라" : 궤적선분을 추출하는 과정

본 논문에서는 각 유형의 검색 질의에 대해서 다음과 같은 네 가지의 질의 셋 유형을 생성하였다. 영역 질의 셋은 두 가지 유형으로 구분하였다. 즉, 모든 시공간 도메인에 대해서 일정한 비율의 크기를 갖는 영역 질의 셋 유형 $Q_R(P)$와 시간 도메인에 대해서는 전체 도메인의 크기에 대한 비율이 아닌, 고정된 크기를 갖는 영역 질의 셋 유형 $Q_R(T,R)$로 구분하였다. 매개변수 P와 R은 전체 도메인 크기에 대한 비율을 의미하며, 매개변수 T는 이동체 위치 보고주기(또는 센서 데이터 수집주기)의 배수를 의미한다. $Q_R(T,R)$을 별도로 정의한 이유는 시간 도메인의 경우에 지속적으로 크기가 증가하기 때문에 전체 시간 도메인에 대하여 일정한 비율로 영역질의의 크기를 정할 경우 동일한 1%의 크기라 하더라도 색인이 생성된 후 얼마의 시간이 지났는가에 따라서 그 크기가 달라지기 때문이다. 예를 들어, 색인이 2006년 1월 1일에 생성되었을 때, "오늘 오후 2시-3시에 롯데 호텔 앞을 지나는 차량을 찾아라."라는 질의를 수행하기 위해 생성되는 영역의 크기는 2006년 2월 1일과 2006년 12월 1일에 동일하지만, 전체 도메인의 크기에 대한 비율은 서로 다르다. 왜냐하면 공간 도메인의 크기는 고정되어 있지만, 시간 도메인의 크기는 시간의 흐름에 따라서 계속해서 커지기 때문이다. 따라서 시간 도메인에 대해서는 비율이 아니, 일정한 크기를 갖는 질의 셋 유형인 $Q_R(T,R)$이 필요하다.

타입 슬라이스 질의 셋 유형 $Q_T(P)$은 시간 도메인에 대해서는 일정한 크기 또는 비율이 아닌, 특정 시간 값을 갖는 질의 집합을 의미하며, 매개변수 P는 전체 도메인 크기에 대한 비율을 의미한다.

복합 질의 셋 유형 $Q_C(I,O)$은 특정 이동체(궤적)를 선택하기 위한 내부영역(Inner Range)과 선택된 이동체(궤적)에 대해서 일부 구간의 궤적선분을 추출하기 위한 외부영역(Outer Range)에 따라서 다양하게 생성할 수 있다. 매개변수 I와 O는 전체 도메인의 크기에 대해서 일정한 비율을 의미한다.

본 논문에서는 각 질의집합에 대해서 질의 결과의 보편성을 보장하기 위해 각각 1,000개의 질의로 구성하였으며, 실험 결과는 모든 질의에 대해서 평균적인 결과를 사용한다. 예를 들면, $Q_R(1)$ 질의 집합은 모든 도메인에 대해서 1%의 크기를 갖는 영역질의 1,000개로 구성되어 있다.

### 4.3 도메인 비율에 따른 성능 비교(데이터셋 유형 D(2,3,S)에 대한 실험 결과)

이 절에서는 전체 색인의 크기가 일정한 상황에서, 전체 도메인에 대해서 일정 비율의 질의영역을 갖는 질의 셋으로 실험한 결과를 보인다. 데이터 셋

<표 6> 영역 질의 및 타임 슬라이스 질의 셋 유형

| 질의 셋 유형 | 시간축 간격 | 공간축 간격 | 비 고 |
|---|---|---|---|
| $Q_R(P)$<br>단, P는 0.05, 1, 20임 | 0.05% | | 시간축 간격: 전체 시간 도메인에 대한 비율(예, 1%의 시간간격) |
| | 1% | | |
| | 20% | | |
| $Q_R(T,R)$<br>단, T는 10 또는 100,<br>R은 0.5, 1, 5, 10임 | 이동체 위치보고주기의 10배 | 0.5% | 시간축 간격: 일정한 크기(예, 10분) |
| | | 1% | |
| | | 5% | |
| | | 10% | |
| | 이동체 위치보고주기의 100배 | 0.5% | |
| | | 1% | |
| | | 5% | |
| | | 10% | |
| $Q_T(P)$<br>단, P는 1 또는 10 | 임의시간 | 1% | 시간축 간격: 간격이 아니라, 특정 시간 (예, 2006년3월1일 11시10분) |
| | | 10% | |

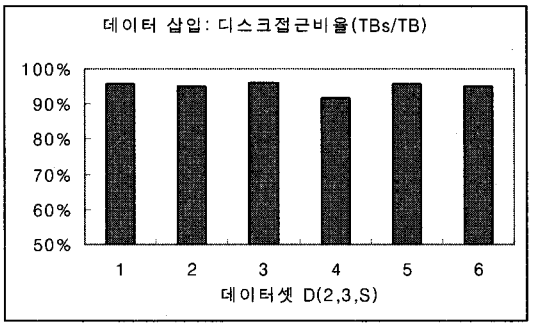
<표 7> 복합 질의 셋 유형

| 질의 셋 유형 | 내부영역 | 외부영역 | 비고 |
|---|---|---|---|
| $Q_R(I, O)$<br>단, I는 0.05, 1, 20,<br>O는 1, 5, 25임 | 0.05% | 1% | 내부영역은 복합질의의 첫 번째 단계인 이동체 선택을 위한 질의영역이며, 외부영역은 선택된 궤적에 대해서 일부 구간의 궤적선분을 추출하는 데 필요한 질의영역임 |
| | | 5% | |
| | | 25% | |
| | 1% | 1% | |
| | | 5% | |
| | | 25% | |
| | 20% | 1% | |
| | | 5% | |
| | | 25% | |

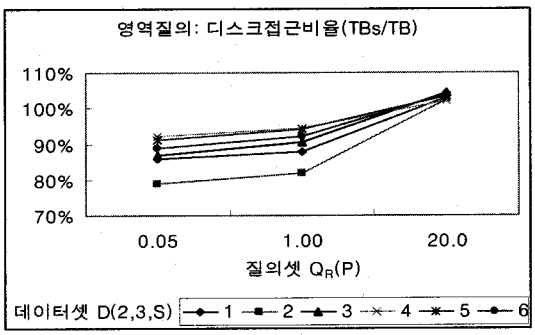
유형 D(2,3,S)을 사용하며, 질의 셋 유형으로는 $Q_R(P)$, $Q_T(P)$, $Q_R(I,Q)$를 사용한다. 실험 결과는 ((TBS-tree의 디스크접근횟수)/(TB-tree의 디스크접근횟수))를 백분율로 표현한다. 즉, 100보다 큰 수치는 본 논문에서 제안한 방법으로 분할된 TBS-tree의 성능이 나쁜 것을 의미하며, 반대로 100보다 작은 경우에는 TBS-tree의 성능이 향상되었음을 의미한다. 본 실험에서 사용된 TBS-tree는 기존 TB-tree를 5개의 요소 색인으로 분할하여 구성하였다.

<그림 5>는 데이터 셋 유형 D(2,3,S)에 대해서 데이터 삽입 성능의 결과를 보이고 있다. 즉, 100개의 이동체와 각 1,000번의 위치 정보에 대해서 총 100,000번의 삽입 연산을 수행한 결과이다. 대부분의 경우에 4-5%의 성능이 개선되었음을 알 수 있다. 특히 시라리오 4의 경우 최대 8%의 성능 향상을 보였다.

<그림 6>은 데이터 셋 유형 D(2,3,S)에 대해서 질의 셋 유형 $Q_R(P)$를 수행한 결과를 보이고 있다. 영역질의의 경우 각 도메인별로 0.05% 비율의 질의영역에서는 8-20%의 비교적 큰 성능 향상을 보였다. 그러나 질의 영역의 크기가 각 도메인별로 20% 비율만큼 커질 경우에는 오히려 성능이 떨어지는 것을 알 수 있다. 왜냐하면, 질의 영역이 커질

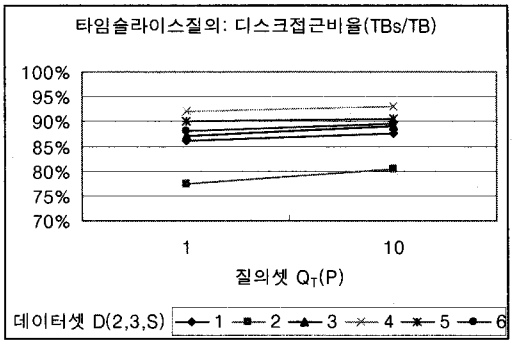



<그림 5> 데이터 삽입 성능, D(2,3,S)




<그림 6> 영역질의 성능, D(2,3,S), $Q_R(P)$

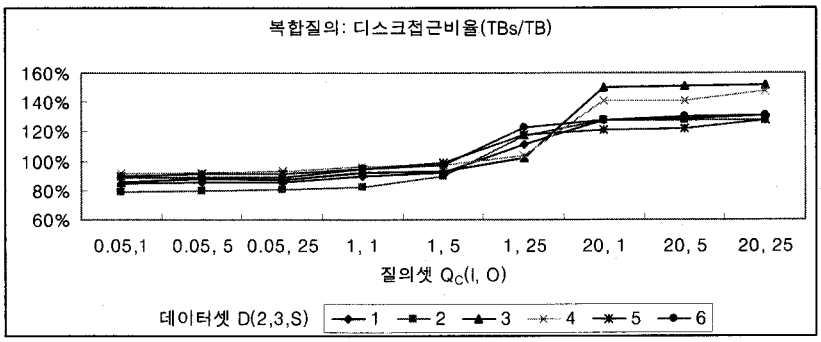
수록 분할된 색인은 여러 개의 요소 색인에 대해서 각각 질의를 수행하고, 각각의 질의 결과를 취합하는 과정이 필요하기 때문이다. 일반적으로 영역질의의 크기는 1% 미만인 경우가 대부분이므로 색인 분할로 인해 영역질의 성능 개선이 기대된다.



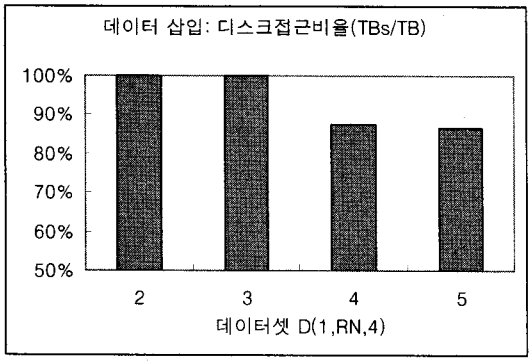

<그림 7> 타임 슬라이스 질의 성능

<그림 7>에서는 타임 슬라이스 질의에 대한 성능 평가 실험 결과를 보이고 있으며, 영역질의의 결과와 매우 유사함을 알 수 있다. 특히 시나리오 2[10]는 시간에 따라 이동체들이 북동쪽 방향으로 방향성을 갖고 함께 움직이기 때문에, 분할되는 요소 색인의 데이터들이 공간적 지역성이 높은 특성을 갖게 됨으로서, 영역 질의 또는 타임 슬라이스 질의에서 우수한 성능을 보이고 있다.

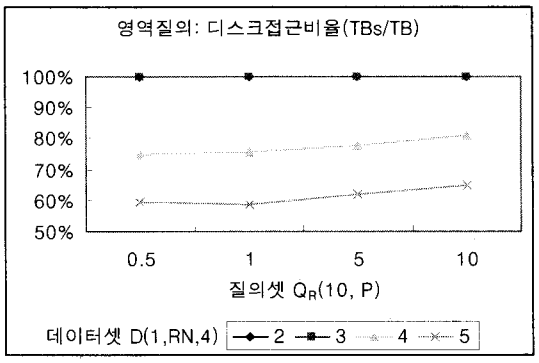
<그림 8>은 데이터 셋 유형 D(2,3,S)에 대해서 질의 셋 유형 $Q_C(I,O)$를 수행한 결과로서, 복합 질의를 구성하는 내부영역은 0.05%, 1%, 20%, 외부영역은 1%, 5%, 25%로 수행하였다. 영역 질의의 결과에서 살펴본 바와 같이 질의 영역이 커질 경우 성능이 떨어지는 것을 보이고 있다. 내부질의 영역이 1%인 경우에도, 외부영역이 25%인 경우 (질의 셋 $Q_C(1,25)$)에도 성능이 기존 TB-tree 보




<그림 8> 복합 질의 성능




<그림 9> 데이터 삽입 성능, D(1,RN,4)




<그림 10> 영역질의 성능, D(1,RN,4), $Q_R$(10,P)

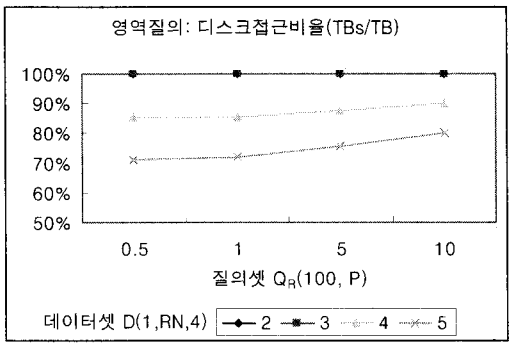
다 떨어지는 것을 보였다. 내부영역 1%인 경우 영역질의만 수행한 경우에는 5-18%의 성능이 개선되었으나(<그림 6> 참조), 궤적 추출 단계에서 25%의 외부영역을 검색해야 하므로 다수의 요소 색인에 대해서 개별적으로 질의를 처리함으로서 성능이 떨어지게 됨을 알 수 있다.

### 4.4 보고 횟수에 따른 성능 비교 (데이터 셋 유형 D(1,RN,4)에 대한 실험 결과)

이 절에서는 전체 색인의 크기가 변하는 상황에서, 시간 도메인에 대해서 일정 크기의 질의영역을 갖는 질의 셋으로 실험한 결과를 보인다. 데이터 셋 유형 D(1,RN,4)를 사용하며, 질의 셋 유형으로는 $Q_R$(T,P)를 사용한다. 실험 결과는 4.3절과 마찬가지로 ((TBs-tree의 디스크접근횟수)/(TB-tree의 디스크접근횟수))를 백분율로 표현한다. 본 실험에서 사용된 TBs-tree는 기존 TB-tree를 데이터 셋에 따라 각각 $10^{RN}/10^3$(단, RN≥3)개의 요소 색인으로 분할하여 구성하였다. 즉, 평균적으로 1,000번의 위치보고마다 하나의 요소 색인을 생성하였다.

<그림 9>는 데이터 셋 유형 D(1,RN,4)에 대해서 데이터 삽입 성능의 결과를 보이고 있다. 즉, 10개의 이동체와 각 $10^2$에서 $10^5$번의 위치 정보에 대해서 삽입 연산을 수행한 결과이다. 요소색인이 1개인 경우에는 성능에 차이가 없으나, 새로운 요소 색인 생길 경우에 성능이 향상되는 것을 보여준다.

<그림 10>과 <그림 11>은 데이터 셋 유형 D(1,RN,4)에 대해서 영역질의에서 시간크기를 이동체의 보고간격의 10배(질의 셋 유형 $Q_R$(10,P)), 100배(질의 셋 유형 $Q_R$(100,P))로 설정해서 실험을 하였다. 데이터 셋 D(1,5,4)에서 가장 좋은 성능을 보임을 알 수 있다. 즉, 색인의 크기가 커질수록 분할된 색인의 성능이 우수함을 보여주고 있다. 왜냐하면, 분할된 색인 TBs-tree에서는 이동체의 보고 횟수가 증가하더라고 각각의 요소색인의 크기는 일정하기 때문에 평균 디스크 접근 횟수가 일정하기 때문이다. 반면에 TB-tree에서는 이동체의 보고 횟수가 증가하면, 전체 색인의 크기가 커지기 때문에, 동일한 크기의 영역 질의에 대해서 시간이 지남에 따라서 성능이 떨어지게 된다.




<그림 11> 영역질의 성능, D(1,RN,4), $Q_R$(100,P)

## 5. 결 론

본 논문에서는 대용량 센서 데이터 아카이빙을 지원하기 위해서 기존 색인을 분할해서 관리할 수 있는 색인 분할 기법을 제안하였다. 그리고 이동체 데이터에 대한 대표적인 색인인 TB-tree를 대상으로 색인 분할 기법을 적용하여, 기존 TB-tree와 분할된 TB-tree(TBs-tree)에 대해서 성능평가 실험을 수행하였다. 실험 결과 색인의 생성비용은 총 100,000번의 삽입 연산에 대해서 최대 8%의 성능 향상을 보였으며, 삽입되는 데이터의 개수가 많아질수록 성능이 더 향상됨을 보였다. 영역질의의 경우 각 질의영역이 적을수록, 시간도메인의 크기가 커질수록 큰 성능 향상을 보였다. 또한 타임 슬라이스 질의 및 복합 질의의 경우에도 영역질의에 대한 실험 결과와 매우 유사한 결과를 보였다. 질의영역의 크기가 전체 도메인의 1% 미만의 범위에서는 색인 분할 기법이 매우 유용하게 적용될 수 있을 것으로 기대된다.

본 논문에서 제안한 시간 분할 색인 방법은 색인을 분할할 때, 일정한 시간 간격을 고정적을 사용하고 있다. 향후, 응용 분야의 특성에 따른 최적의 시간 간격을 결정하기 위한 방법과 가변적인 시간 간격을 고려한 색인 분할 방법에 대한 연구가 필요하다. 또한 시간 분할 색인 방법을 활용하여, 아카이빙된 데이터의 활용 방안에 대한 연구도 추가로 필요하다.

## 참고문헌


1. S. Jang, and W. Woo, "ubi-UCAM: A Unified Context-Aware Application Model," LNAI/LNCS, vol 2680, 2003, pp. 178-189.
2. Marios Hadjieleftheriou, George kollios, Dimitrios Gunopulos, Vassilis J. Tsotras, "Efficient Indexing of Spatiotemporal Objects", Proceedings of Extending Database Technology, 2002, pp. 251-268.
3. Mario A. Nascimento, and Jefferson R. O. Silva, "Towards Historical R-trees", Proceedings of ACM Symposium on Applied Computing, 1998, pp. 235-240.
4. Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis, "Novel Approaches to the Indexing of Moving Objects", Proceedings of International Conference on Very Large Data Bases, 2000, pp. 395-406.
5. Hongjun Zhu, Jianwen Su, and Oscar H. Ibarra, "Trajectory Queries and Octagons in Moving Object Databases", Proceedings of the ACM Conference on Information and Knowledge Management, 2002, pp. 413-421.
6. Antonm Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of ACM-SIGMOD Conference on the Management of Data, pp. 47-57, 1984.
7. Yannis Theodoridis, Michael Vazirgiannis, and Timos Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications", IEEE International Conference on Multimedia Computing and Systems, 1996, pp. 441-448.
8. Yufei Tao, and Dimitris Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries", Proceedings of International Conference on Very Large Data Bases, 2001, pp. 431-440.
9. Mohamed F. Mokbel, Thanaa M. Ghanem, and Walid G. Aref, "Spatio-temporal Access Methods", IEEE Data Engineering Bulletin, Vol.26 No.2, 2003, pp. 40-49.
10. Yannis Theodoridis, Jefferson R. O. Silva, and Mario A. Nascimento, "On the Generation of Spatiotemporal Datasets",

Advanced in Spatial Databases, 1999, pp. 147-164.

11. Thomas Brinkhoff, "A framework for Generating Network-Based Moving Objects," GeoInfomatica vol.6(2), 2002, pp. 153-180.
12. Michalis Vazirgiannis, Yannis Theodoridis, and Timos K. Sellis, "Spatio-Temporal Composition and Indexing for Large Multimedia Applications," Multimedia Systems 6(4), 1998, pp. 284-298.

**조대수**

1995년 부산대학교 컴퓨터공학과 졸업(공학사)
1997년 부산대학교 컴퓨터공학과 졸업(공학석사)
2001년 부산대학교 컴퓨터공학과 졸업(공학박사)
2001년~2004년 한국전자통신연구원
텔레매틱스연구단 선임연구원
2004년~현재 동서대학교 컴퓨터정보공학부 조교수
관심분야 : GIS, 공간데이터베이스, LBS, 스트림 데이터처리 등