LBS 엔진개발을 위한 Moving Objects DB 기술 연구

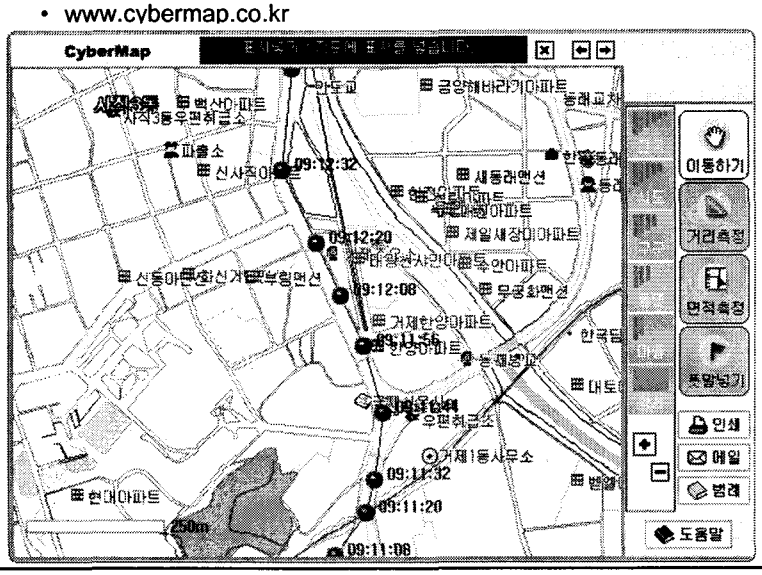
Hong Bong Hee
Department of Computer Engineering
Pusan National University
Email : bhhong@pusan.ac.kr

Contents

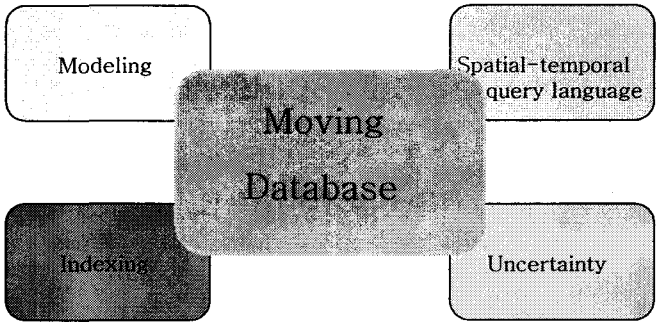
- LBS and Moving Objects databases
- Indexing of Moving Objects databases
- New approaches to spatio-temporal indexing
- Open issues and summary

Tracking services of moving objects

- Web-based tracking of moving objects
  - www.cybermap.co.kr



Moving Objects Databases(1/3)



[Figure 1] Research area of Moving Objects DB

Moving Objects Databases (2/3)

- Location Modeling
  - Modeling of continuously moving locations
  - Modeling approaches
    - Points in 3 dimensional space
    - line segment in 3 dimensional space
    - dynamic attribute with time function

1-5

Moving Objects Databases (3/3)

- Indexing
  - Indexing of movement of continuously changing location
  - Requirements
    - Frequent updates of locations
    - Finding out the past/current/future location
  - example
    - Current/future
      - TPR-tree
    - Past
      - 3DR-tree, STR-tree, TB-tree,

1-6

Moving objects DB의 필요성

- LBS의 성능은 moving objects DB에 따라 좌우됨
  - Scalability
    - moving objects의 수가 수천 ~ 수백만으로 증가
    - moving objects trajectories의 크기 = moving objects 수 x 위치 보고 횟수
- Characteristics of moving objects DB
  - Frequent and continuous updates of location
    - 대량의 Moving Objects에 대한 동시 다발로 발생하는 update transaction
      - index 및 데이터에 대한 실시간 update 필요
  - Queries on trajectories, current positions, and future positions
    - 3D indexing and dynamic attributes

Spatial Access Methods (SAM)

- Spatial Access Methods (**SAM**)
  - Space-driven structures
    - 2 dimensional plane에서 objects의 분포와는 무관하게 2D space를 격자 구조나 사분구조로 분할하여 indexing
      - Grid file, Quad-tree
  - Data-driven structures
    - object group에 대한 MBR(Minimum Bounding Rectangle)으로 search space을 분할하는 index 구조
      - R-tree, R*-tree, R+-tree

1-8

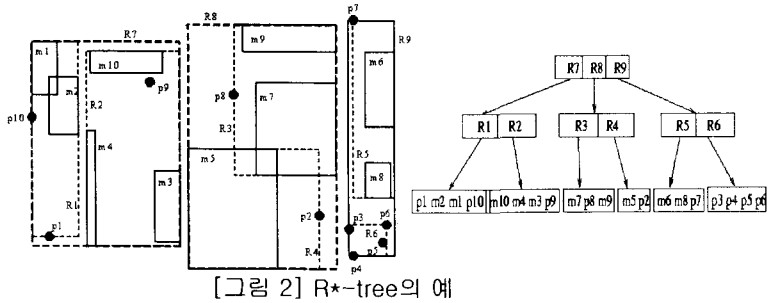
Spatial Access Methods (SAM)

- moving objects index 관점에서의 SAMs
  - Space-driven structures: 그리드 파일
    - moving objects의 분포는 비균등 분포
      - 차량 정체 지역의 경우 object가 집중된다.
      - Space-driven index의 성능 저하
    - moving objects의 분포는 시간이 지남에 따라 바뀐다.
      - 동적인 index 구조의 필요성
    - 시간 domain은 시간이 지날 수록 성장하기 때문에 moving objects index에서는 space-driven structures가 부적합하다.
  - Data-driven structures: R-tree
    - object 분포를 중심으로 분할하는 index 구조
    - 동적으로 입력되는 3D index 구조로 적합

1-9

SAM : R*-tree (1/2)

- R*-tree [Beck90] 는 R-tree 와 동일한 index 구조
- R-tree의 문제점
  - node의 MBR들 사이의 Overlap 문제로 탐색 성능이 저하된다.
- R*-tree의 objectives
  - the overlap between directory rectangles should be minimized.
  - Storage Utilization should be maximized.

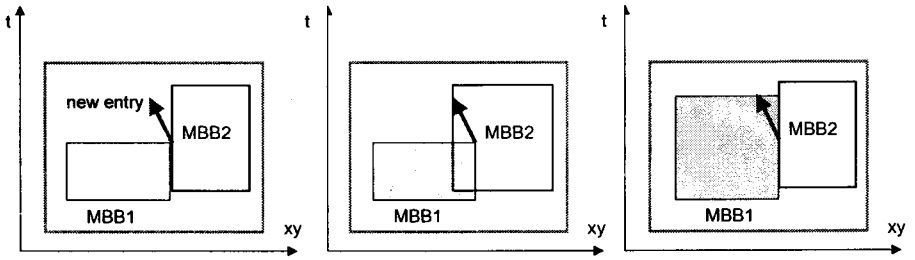


[그림 2] R*-tree의 예

1-10

SAM : R*-tree (2/2)

- ChooseSubtree 알고리즘
  - Choose Best Leaf Node
    - determine the minimum overlap enlargement
  - Choose Best Non-Leaf Node
    - determine the minimum area enlargement



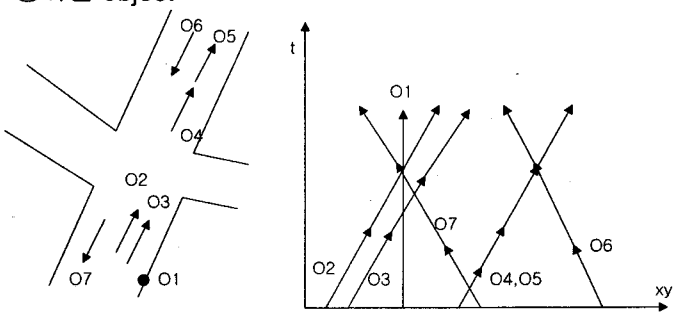
(a) 새로운 entry의 삽입 (b) Minimum Area Enlargement (R-tree) (c) Minimum Overlap Enlargement (R*-tree)

[그림 3] R-tree와 R*-tree의 ChooseSubtree의 차이

1-11

기존 spatial index의 문제점 (1/2)

- moving objects의 이동 특징을 반영하지 못함
  - 같은 도로를 주행하는 moving objects의 이동은 중복 MBB 발생
  - 정지 object, 같은 방향으로 이동하는 object, 반대 차선에서 이동하는 object



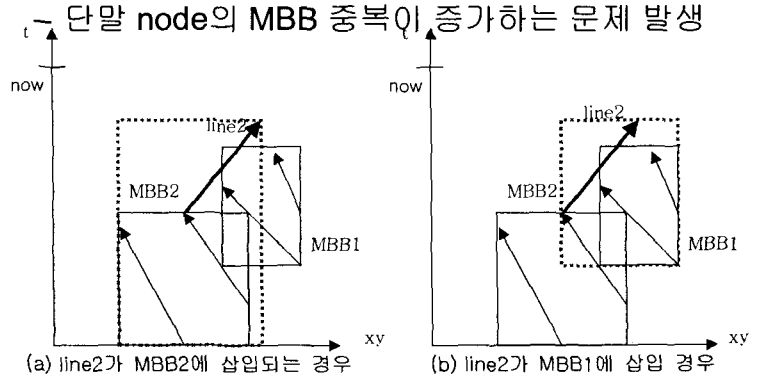
(a) 도로상의 이동체 (b) 시간 도메인을 고려한 이동체

[그림 4] moving objects의 이동 특징

1-12

기존 spatial index의 문제점 (2/2)

- 이동 trajectories의 삽입 특징으로 인한 문제점 발생
  - 시간, spatial 축으로 MBB Enlargement
  - 단말 node의 MBB 중복이 증가하는 문제 발생



[그림 5] R*-tree의 ChooseSubtree의 문제점

1-13

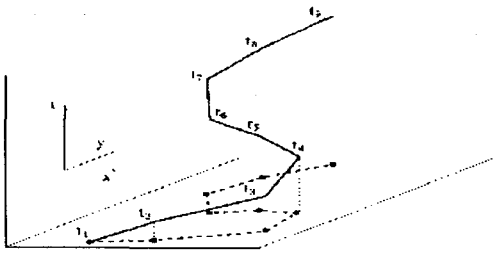
최근 moving objects index 기술

- trajectories 질의 index(TB-tree)
- 현재 위치 index(Hash-based)

1-14

trajectories 질의 index(TB-tree)(1/3)

- TB-Tree(Trajectory Bundle Tree) [PJT00]
  - *modeling*
    - moving objects의 trajectories: **Line segment**의 집합
  - moving objects의 trajectories 보존에 중점
    - 동일한 object의 trajectories은 같은 node에 저장
  - 장점
    - 특정 moving objects의 **trajectory query** 성능 우수
  - 단점
    - *moving objects*의 *spatial*적 특성 고려하지 않음
      - 단말 node간의 overlap이 심하여 영역 질의 성능이 낮음.
      - 삽입 연산을 위한 FindNode()의 비용 증가로 삽입 성능이 나쁨.

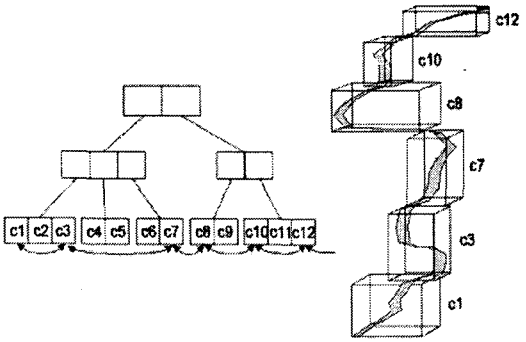


[그림 6]moving objects의 이동과 trajectories

1-15

trajectories 질의 index(TB-tree)(2/3)

- index 구조
  - 단말 node에 하나의 trajectories만 저장
  - Trajectory Preservation
    - 같은 moving objects의 trajectories을 Linked List로 연결

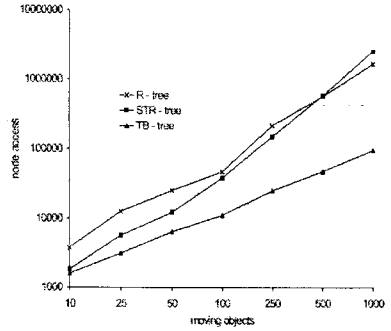


[그림7]Leaf Node가 연결된 TB-Tree의 구조

1-16

trajectories 질의 index(TB-tree)(3/3)

- 특성
  - 삽입시 spatial적 특성을 고려 하지 않음으로 일반적인 spatial-temporal range query에 취약함
  - Trajectory query에서 좋은 성능을 보임



[그림8] Combined Query 수행 결과

1-17

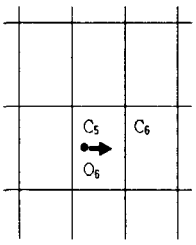
현재 위치 index(Hash-based)(1/4)

- 해쉬 기반의 index 구조
  - 이동object를 point으로 표현
  - 현재 위치 검색만 가능
  - cell 내의 위치 변경은 index의 변화가 없다.
  - 해쉬 함수를 이용한 빠른 검색 및 변경 연산 가능

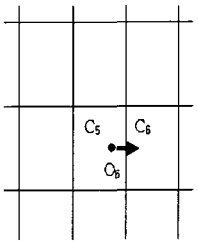
1-18

현재 위치 index(Hash-based)(2/4)

- 해쉬 기반의 index 구조에서의 위치 변경
  - 셀 내에서의 이동
    - index 구조에서의 변경은 없다.
  - 다른 셀로의 이동
    - C5 셀에서 삭제, C6 셀에서 삽입
    - O6의 이전 좌표와 현재 좌표를 이용하여 해당 셀을 간단히 검색



[그림9] 셀 내에서 이동

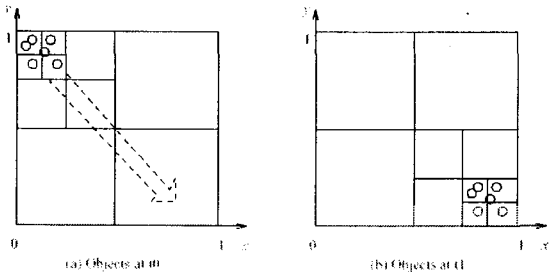


[그림10] 다른 셀로 이동

1-19

현재 위치 index(Hash-based)(3/4)

- **[SR01]**은 **Quad-Tree**를 **index**으로 사용
  - 단점
    - moving objects의 이동이 빈번한 경우에 셀을 분할하고 병합하는 비용이 높다.
  - 시간의 흐름에 따라 계속적인 이동 object의 삽입, 삭제를 발생시키는 동적 구조를 지원하는 index 구조가 필요.



[그림11] moving objects의 Skewed distribution

1-20

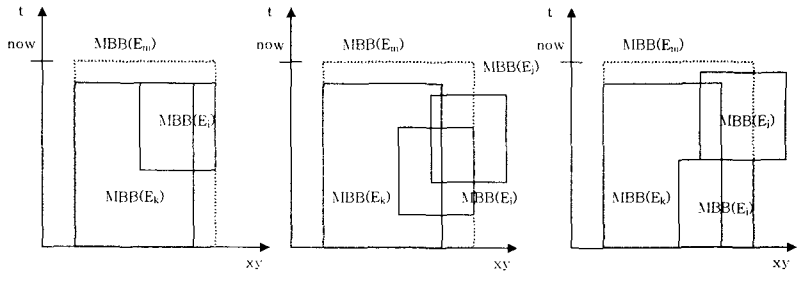
moving objects index의 새로운 접근 방법

- moving objects index의 특징
  - node간의 overlap이 심하다.
  - node의 dead space가 크다.
  - 시간 축으로 증가하는 time domain을 고려해야 한다.
- Solution
  - The Forced Merging Policy
  - The clipping policy
  - The Unbalanced Splitting policy
- Unified Indexing의 필요성
  - moving objects index에 관련된 기존 index 연구는 현재 위치와 과거 위치를 구분하여 index 구조 제안
  - Current Position time-based R-Tree
    - moving objects의 과거 및 현재 위치 검색을 지원

1-21

The Forced Merging Policy

- To reduce the overlap between nodes
  - HOR (Highly Overlapping Rate)
  - HOMbb (Highly Overlapped MBBs)
  - The cascading of HOMbb



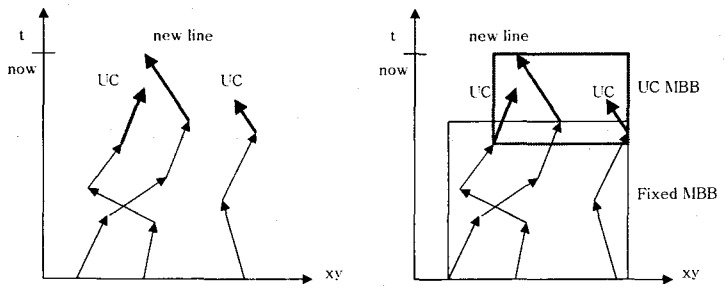
(a) 단순한 강제 합병 (b) 공간 축으로 HOMbb의 연쇄 (c) 시간 축으로 HOMbb의 연쇄

[그림12] 강제 합병의 예

1-22

The Unbalanced Splitting policy

- The Unbalanced Splitting Policy along time axis
  - To increase the storage utilization
  - UC line (Until Changed line)
  - UC MBB (UC minimum bounding box)
  - Fixed MBB



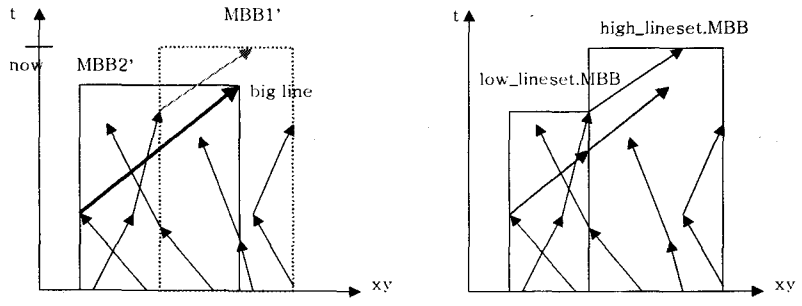
(a) 새로운 선분의 삽입으로 오버플로우 발생 (b) 시간 축의 비균등 분할

[그림13] 비균등 분할의 예

1-23

The clipping policy

- To reduce the dead space of nodes
  - The big line segment should be defined by related size of splitting nodes



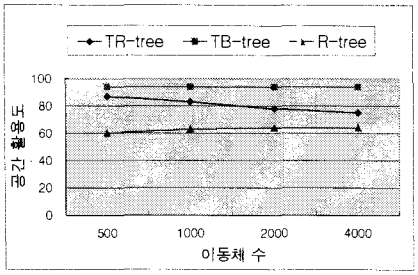
(a) 2개로 분할된 MBB들이 HOMbb인 경우 (b) 2개의 선분으로 절단하여 삽입

[그림14] Big line segment의 clipping 예

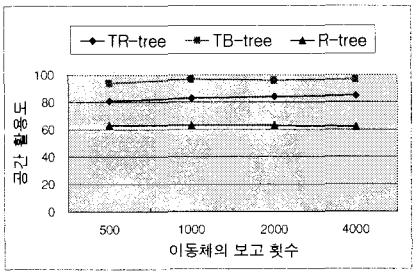
1-24

Performance Studies (1/2)

- Datasets
  - is generated using CitySimulator V2.0
- The storage Utilization
  - TB-tree is best, because same trajectories is sequentially stored in leaf



(a) 이동체 수의 증가에 따른 공간 활용도

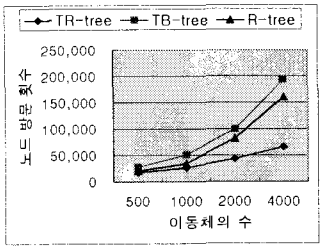


(b) 이동체 보고횟수의 증가에 따른 공간 활용도

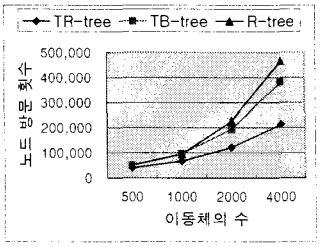
1-25

Performance Studies (2/2)

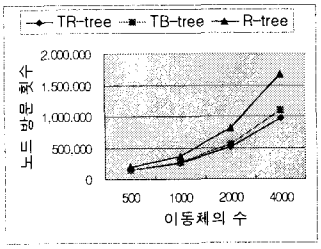
- Range Query
  - 3DR-tree is good in small range
  - TB-tree is good in large range
  - TR-tree is best in all case



(a) RQS = 5%



(b) RQS = 10%



(c) RQS = 20%

1-26

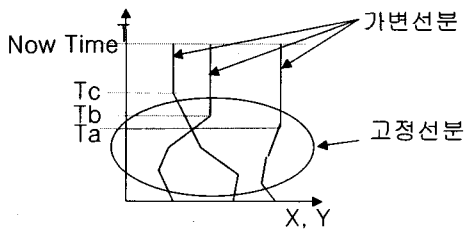
Current Position time-based R-Tree의 모델링 (1/2)

- moving objects의 현재 위치
  - 보고시간의 갭(Gap)으로 인하여 정확한 현재 위치를 알 수 없음
  - moving objects가 가장 최근에 보고한 spatial상의 좌표를 moving objects의 현재 위치로 정의한다.
- moving objects의 과거 위치
  - 인접한 보고 시점의 점들을 이은 선분으로 표현

1-27

Current Position time-based R-Tree의 모델링 (2/2)

- index 상의 moving objects 위치 표현 방법
  - 현재 위치
    - 가변선분: 선분을 나타내는 좌표에 Now Time을 가짐
  - 과거 위치
    - 고정 선분: 선분을 나타내는 좌표에 Now Time이 아닌 다른 시간 값을 가짐

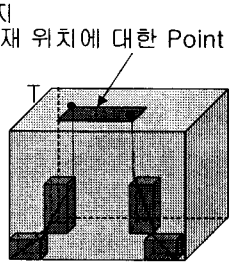


[그림15] 현재 위치 및 과거 위치 표현 방법

1-28

Current Position time-based R-Tree의 자료구조

- 현재 위치 검색을 위한 Point MBR
  - 현재 위치
    - 시간이 Now Time인 2차원 점
  - 현재 위치에 대한 Point MBR을 유지함
    - 단말node에 가변선분이 있을 경우 그 상위node에 Point MBR유지
  - 과거 위치
    - line MBR 유지

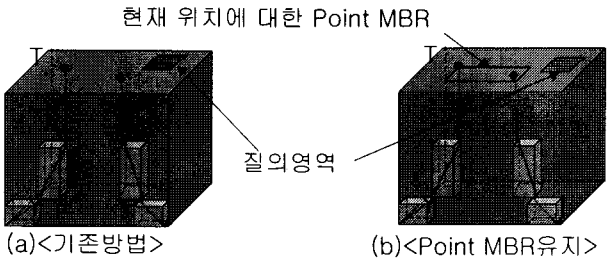


[그림16] 현재 위치에 대한 Point MBR

1-29

Current Position time-based R-Tree의 검색 알고리즘

- 검색 알고리즘
  - 현재 위치 검색
    - Point MBR을 이용하여 검색
    - Search Space를 줄임으로서 검색성능 향상
  - 과거 위치 검색
    - Line MBR을 이용하여 검색
    - R-Tree 검색 방법과 동일



[그림 17] moving objects 위치 검색

1-30

Summary and Open issues (1/3)

- moving objects DB의 modeling and indexing
  - MOST(Moving Objects Spatial Temporal)
    - Dynamic attributes
  - Spatio-Temporal Data Model
    - Trajectories indexing
- Effects of existing works on indexing of moving objects databases
  - TPR-tree : search current & future location
  - STR-tree, TB-tree : search trajectories of moving object
  - HR-tree, MV3R-tree : perform time-slice query efficiently
  - 3DR-tree : perform range query efficiently

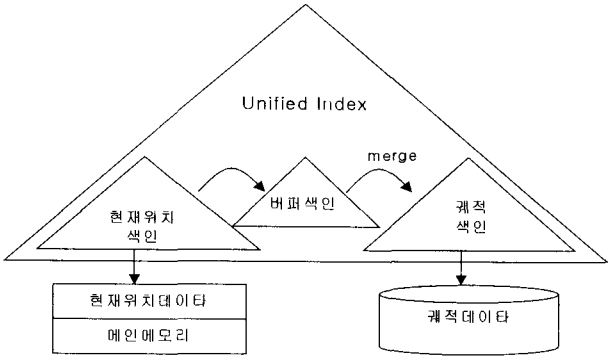
Summary and Open issues(2/3)

- Open issues and problems
  - Unified indexing on trajectories and current positions
    - Should be able to process Now or Until Changed
    - Attempt to combine trajectories index with current position index
  - Need to overcome the problems of R-tree based indexing
    - Reduce Dead space and overlapped MBR
    - Attempt to develop a new version of 3D R*-tree

1-32

Summary and Open issues (3/3)

- Open issues and problems
  - Unified indexing on main-memory DB and disk-based DB for storing the movement of moving objects



[그림18] Unified index structure

1-33

참고 문헌

- [Beck90] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," ACM SIGMOD Conference, pp.322-331, 1990.
- [PJT00] D. Pfoser, C. S. Jensen, Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects", Proc. Of the 26th VLDB Conf., Cairo, Egypt, September 2000.
- [SJLL00] S. Saltenis, C. S. Jensen, S.T. Leutenegger, and M. A. Lopez, "Indexing the Positions of Continuously Moving Objects.", Proc. ACM SIGMOD on Management of data 2000.
- [SR01] Zhexuan Song, Nick Roussopoulos: Hashing Moving Objects. Mobile Data Management 2001: 161-172.

1-34