스케줄링 기법을 이용한 분산 이동 객체 데이타베이스의 범위 질의 처리†

Range Query Processing of Distributed Moving Object Databases using Scheduling Technique

전세길*, 황재일*, 나연묵**

Se-Gil Jeon, Jae-Il Hwang, Youn-Mook Nah

요 약 최근 들어 이동 통신 서비스에서 이동하는 고객의 위치정보와 관련된 서비스가 중요한 서비스로 부각되고 있다. 이동 객체의 경우 갱신 연산이 많고 부하가 특정 지역에 집중되는 특징이 있다. 이러한 LBS 응용에서 시공간 영역 질의는 중요한 서비스이다. 질의 처리 속도의 개선을 위해서 디스크 입출력 시간을 최대한 줄일 필요가 있다.

본 논문에서는 갱신연산을 최소화하기 위해 고안되어진 시공간 시스템 아키텍쳐로 GALIS의 비균등 2단계 격자 인덱스 구조를 적용한다. 질의 처리 처리율을 향상시키기 위해서 공간 관련성과 시간 관련성을 이용한 스케줄링 기법과 Time Zone 개념을 사용하여 시공간 개념이 결합된 질의 처리 기법을 제안한다. 제안된 방법의 성능 측정을 위해서 다른 질의 범위를 가지고 영역 질의의 결과를 보인다.

Abstract Recently, the location-based service for moving customers is becoming one of the most important service in mobile communication area. For moving object applications, there are lots of update operations and such update loads are concentrated on some particular area unevenly. The primary processing of LBS application is spatio-temporal range queries. To improve the throughput of spatio-temporal range queries, the time of disk I/O in query processing should be reduced.

In this paper, we adopt non-uniform two-level grid index structures of GALIS architecture ,which are designed to minimize update operations. We propose query scheduling technique using spatial relationship and time relationship and a combined spatio-temporal query processing method using time zone concepts to improve the throughput of query processing. Some experimental results are shown for range queries with different query range to show the performance tradeoffs of the proposed methods.

주요어 : 위치 기반 질의 처리, 위치 기반 서비스, GALIS, 질의 스케줄링

Keywords : Location-based Query Processing, Location-based Services, Query Scheduling

1. 서 론

최근 들어 위치 기반 서비스(Location Based Service : LBS) 분야에서 대량의 이동 객체의 위치 정보를 실시간으로 관리해야할 필요성이 증가 하고 있다. 일반적으로 위치 기반 서비스는 3가지 중요한 기술이 있다. 위치를 결정하는 기술, 위치 응용 플랫폼, 위치 응용 서비스이다. 대부분의 위치 기반 서비스가 시간, 공간 모두와 관련이 있으며, 각 시간대 별로 위치 정보를 저장하게 된다. 즉, 저장 대상이 이동 객체가 되며 질의 형태도 기존의 지리정보시스템(Geographic Information System)에서와 다르게 시간적 특성과 공간적 특성 모두를 이용하는 시공간 질의의 형태가 대부분이다. 이동객체의 특징은 크게 두 가지로 볼 수 있다. 첫째 연속적으로 이동하기 때문에 빈번한 갱신 연산이 수행된다. 서비스 가입자의 위치 변경은 인덱스 구조의 변경을 초래하기 때문에 인덱스 구조가 자주 변하지 않는 구조가 적합하다. 둘째 핸드폰 가입자의 경우 도시의 중심가에 특정 시간에 밀집되는 현상이 발생하는 특징이 있다.

† 본 연구는 대학 IT연구센터 육성지원 사업의 지원에 의해 수행 되었음

* 단국대학교 대학원 전자컴퓨터 공학과 {sgjeon@dankook.ac.kr, hwangjaeil@yahoo.co.kr}

** 단국대학교 전기전자 컴퓨터공학부 부교수 ymnah@dku.edu

위와 같은 이동 객체의 위치 정보 처리시의 문제점을 해결하기 위해서 빈번하게 발생하는 갱신연산을 줄이고 대량의 데이타를 효율적으로 처리하기 위해 GALIS 시스템을 사용한다. 특정 시간에 밀집되는 범위 질의의 경우 질의간의 중복되는 데이타를 공유하는 경우가 많으며 불필요한 중복 연산을 줄이기 위해서 질의 스케쥴링 기법을 제안한다. GALIS(Gracefully Aging Location Information System)라 명명된 시스템 구조는 여러 개의 프로세서 노드로 구성되는 클러스터 기반 분산 컴퓨팅 시스템 구조이다 [4][5][6]. 각 노드는 시공간적인 위치 데이타를 공간적인 구역(cell)과 시간적인 구역(zone)을 분할하여 저장한다. 된다. 본 논문에서는 이동 객체의 빈번한 갱신을 저비용으로 처리하기 위한 인덱스 구조로 비균등 2단계 그리드 구조를 사용하며 첫 번째 단계는 크기가 유동적이며 두 번째 단계에서는 고정 그리드 형태이다. 고정 그리드에서 각 셀 간의 순서를 결정하는데 Z-순서(Z-ordering) 기법을 사용 한다. 시공간 데이타베이스에서 대해서 발생하는 질의의 유형은 특정 시간 내에서 특정 객체가 움직인 궤적을 질의 하는 경우와 특정 시간 내에서 특정 지점에 머문 객체들을 구하는 질의가 있다. 또한 특정 시간 범위에 특정 영역내에 있었던 객체들을 구하는 질의가 있을 수 있으며, 이를 범위 질의라 한다. 범위 질의는 사각형이나 원형 형태이며 각 범위 질의 간에는 유사한 겹침 영역이 발생할 수 있다. 본 논문에서는 과거 궤적에 대한 범위 질의를 대상으로 하였으며 논문에서 사용하는 비균등 2단계 그리드 구조의 경우 과거 궤적에 대한 필터링 효과로 인해 일부 데이타가 손실될 수 있으며, 객체의 위치, 속도, 방향을 이용한 다양한 위치 예측 기법으로 해결하는 것으로 가정하였다.

동일한 영역에 대한 질의 간에는 질의 결과를 공유할 수 있으며 같은 데이터에 대한 디스크 재검색 시간을 줄임 으로써 질의 처리율을 향상 시킬 수 있다. 질의 처리 큐에 입력된 질의에 대해서 유사한 영역을 함께 처리하기 위해서는 질의 우선순위를 재조정 할 수 있는질의 스케쥴링 기법이 필요하다.

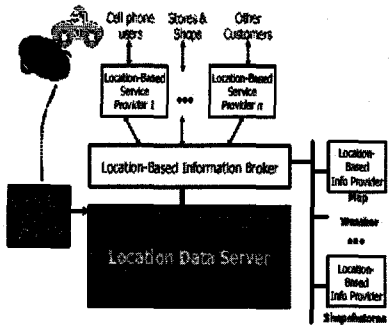
본 논문에서 제안한 스케쥴링 기법은 질의 간에 공간적으로 겹치는 셀의 개수에 따른 공간 관련성과 각 질의에 대해서 시간 범위가 겹치는 정도를 나타내는 시간 관련성을 적용한다. 또한 다중 노드를 가정 하였으므로 질의와 각 노드간의 겹침 관계를 적용한 노드 관련성도 적용한다. 본 논문에서 제안한 질의처리 스케쥴링 기법의 유용함을 보이기 위해 기존의 이동 객체 관련 분야에서 시뮬레이션 할때 범용적으로 사용되는 GSTD(Generate Spatio Temporal Data)의 확장된 데이타 그리고 사용자 패턴을 적용한 데이타를 이용하여 실험 평가 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 기술하고, 3장에서는 시공간 위치 데이타의 처리를 위한 GALIS 구조에 대해서 설명한다. 4장에서는 본 논문에서 제안하는 시공간 질의 처리 기법에 대해서 기술하고 5장에서 성능을 보이고 마지막으로 6장에서 결론과 향후 연구 과제를 제시한다.

2. 관련연구

2.1 위치 기반 서비스

<그림 1>은 크게 5 가지 부분으로 나누어진 기본적인 위치기반 서비스 시스템의 전체 구조를 나타내고 있다. 위치 시스템(Positioning System: PS)은 이동 객체의 현재 위치를 측정하기 위해 사용한다. 위치 데이타 서버(Location Data Server: LDS)는 메인 메모리 기반과 디스크 기반 데이타베이스를 이용하여 이동 객체의 현재, 과거 위치를 저장하고 인덱싱하기 위해 사용한다.



<그림 1> 기본적인 LBS 시스템 구조 [6]

위치 내용 서버(Location Content Server: LCS)는 LDS에서 제공하는 사용자의 위치 정보와 위치기반 정보 제공자(LBIPs)들이 제공하는 여러 가지 지리적 정보를 조합하여 요청된 정보를 실시간으로 위치 서비스 제공자(Location Service Provider: LSP)에게 제공한다. 위치기반 정보 제공자(Location-Based Information Providers: LBIPs)는 LDS에서 제공하는

위치 정보에 추가될 수 있는 지도, 날씨, 상점 등의 개별적인 정보를 제공한다. 위치 서비스 제공자는 모바일 폰 가입자나 상점 등에 위치 정보를 제공하는 독립적인 정보제공자들이다.

LBS 시스템을 사용하는 예를 들어보면 어떤 모바일 폰 사용자가 현재 자기로부터 가장 가까운 친구를 찾는 경우 LSP에게 정보를 요청하게 되고, LSP는 LBIB와 LDS의 정보를 조합하여 저장한 LCS를 이용하여 사용자에게 해당 정보를 제공한다.

2.2 이동객체의 질의처리 실험을 위한 시뮬레이터

2.2.1 GSTD와 확장된 GSTD

GSTD(Generate Spatio Temporal Data) 알고리즘은 시공간 데이타 집합을 생성하는데 있어서 대표적인 알고리즘으로 위치 데이타 생성에서 고려되어져야 할 여러 가지 사용자 파리미터들과 알고리즘을 제시한다[21]. 시간에 따라 그 위치가 변화하는 이동 객체들을 제어하는 파라미터 집합의 정의는 GSTD 알고리즘의 가장 중요하면서도 기본적인 문제이다. 파라미터 집합은 그 기능에 따라 다음의 세 가지 분류로 나눠볼 수 있다.

<표 1> GSTD 알고리즘의 사용자 파라미터 분류

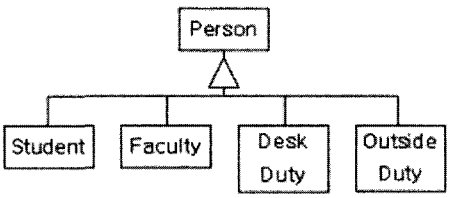
1	객체 인스턴스 시간 간격
2	객체의 위치 이동
3	객체 크기의 재조정

객체의 이동은 시간 간격(duration)을 가지며 매 시간마다 공간 위치의 변화(shift)를 수반한다. GSTD 알고리즘은 이동 객체의 두 가지 형식 이동 점 객체와 이동 영역 객체를 생성할 수 있는데 이동 영역 객체는 시간의 흐름에 따라 그 공간 위치뿐만 아니라 공간 영역의 크기도 사용자가 지정한 수치 내에서 변화하게 만들 수 있다.

확장된 GSTD에서는 GSTD의 알고리즘을 확장하고 사용자 파라미터를 추가하여 사용자가 직접 시간 전개 값과 시간 한계 값을 지정할 수 있도록 되어있고 작업 공간을 확장하여 객체들의 움직임이 하나의 작업 공간을 벗어나 인접한 다른 작업 공간으로 넘어가더라도 객체의 움직임을 생성할 수 있도록 알고리즘이 수정 되었다. 즉 객체의 위치정보 갱신 시간, 객체가 움직이는 시작 시간, 객체가 움직임을 멈추는 시간을 사용자가 직접 지정할 수 있도록 하였고, 객체가 하나의 작업 공간을 벗어나더라도 인접한 다음 작업 공간에서 계속 활동하거나 처음에 생성된 작업 공간 안에서 계속 활동하도록 알고리즘이 수정 되었다.

2.2.2 사용자 패턴을 적용한 시뮬레이터

본 논문의 실험 평가를 위해 서로 다른 움직임 시나리오를 갖는 이동 객체를 표현하기 위해서 데이타 생성기의 이동 객체를 5가지 종류로 분류된다[20].



<그림 2> 이동 객체의 분류[20]

Person 클래스는 내부적으로 이동 객체를 표현하기 위한 최소 정보인 OID, 현재 위치 그리고 GALIS를 위한 Time Interval 변수, VP와의 거리, 과거 위치 정보 및 기타 변수들과 움직임을 결정해주는 함수와 객체의 위치정보를 해당 셀 번호로 계산해 주는 함수를 갖는다. Person 클래스의 움직임은 일정한 시나리오에 따르지 않고, 단지 랜덤하게 그 진행방향 및 움직임 거리가 결정되는 방식으로서 실제 현실에서의 이동 객체의 움직임을 나타내기엔 부적절한 것이다.

```
Person
  int Oid; // 객체의 OID
  POINT Cur_P; // 객체의 현재 좌표
  POINT Old_P; // 객체의 이전 좌표
  int TimeInterval; // 경과 시간 값
  int Cell_No; // 객체의 소속 셀 번호
  int VP1_DIST;       // VP1 과의 거리
  int VP2_DIST;       // VP2 와의 거리
  int Old_Cell_No; // 객체의 이전 셀 번호
  int TimeZone; // GALIS의 TimeZone 값
  int Reserve1; // 예약값 1
  int Reserve2; // 예약값 2
  int MovingMethod; // 객체의 움직임 방법
  int Direct; // 객체의 움직임 방향
int Cal_Cell_No(); // 객체의 셀 번호 계산
bool obj_move(); // 객체의 움직임 결정
```

<그림 3> 사용자 패턴 생성을 위한 Person Class[20]

<그림 3>에서와 같이 데이타 생성기는 우선적으로 이동 객체에 필요한 기본적인 부분만을 구현하여 Person 클래스란 부모클래스를 정의하였다. Person 클래스는 이동 객체를 구성하는 데 필요한 최소한의 변수 및 함수를 구현한 클래스로서 다른 모든 이동 객체 클래스들은 이 클래스를 상속받게 된다.

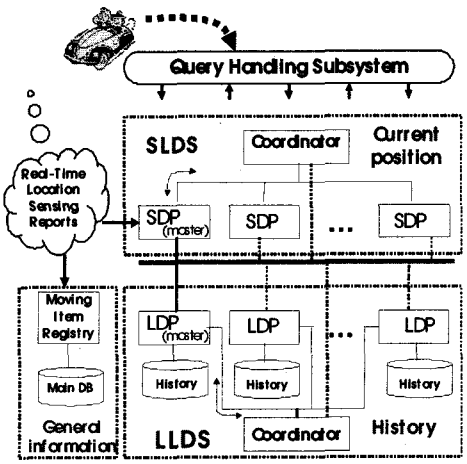
이런 이유로 현실과 유사한 움직임 시나리오를 갖는 자식 클래스들이 필요하게 된다. 본 논문에서 사용하는 데이타 생성기에서는 이러한 움직임을 갖는 이동 객체들의 예로서 Student, Faculty, DeskDuty, OutsideDuty의 4가지 클래스를 정의하였으며, 이러한 모든 자식 클래스들은 부모클래스인 Person 클래스를 상속받는다.

2.3 기존의 시공간 인덱스 구조

기존의 시공간 객체의 위치정보를 인덱싱하기 위한 구조로 기존에 제안된 RT-트리, 3D R-트리, TB-트리, TPR-트리등이 있다[8, 9, 10]. 이러한 기존의 R-트리 기반 구조들은 대량의 이동객체의 움직임에 따라서 많은 노드의 분할 및 합병이 일어난다. 트리에서 노드의 분할, 합병은 삽입, 삭제 시 디스크에 많은 부하를 초래하며 처리 시간도 기하급수적으로 증가한다. 본 논문에서 사용하는 비 균등 2단계 고정 그리드 구조는 이동 객체의 이동정보를 저장 시에 트리 기반 인덱싱 에서의 문제점을 해결하기 위해 고정 그리드를 기반으로 한 2단계의 인덱싱 구조이다

3. GALIS 구조

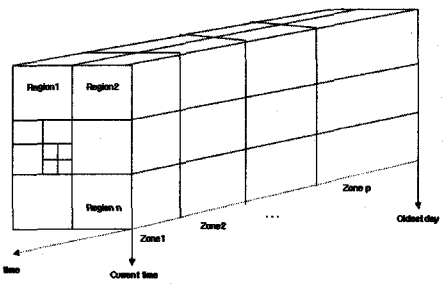
<그림 4>는 GALIS 구조를 나타낸 것이다. SLDS(Short-term Location Data Subsystem)는 이동 객체들의 현재 위치정보를 주관하는 SDP(Short-term Data Processor)와 SDP 노드간의 균형을 유지하기 위한 조정기(Coordinator)로 구성되어있으며, 클라이언트로부터의 질의를 받아서 처리하는 기능을 담당한다. SLDS내의 각 SDP들은 하나의 데이타 프로세서에 대응되고, 메인 메모리 데이타베이스로 구성되어 있으며, 이동객체의 현재 위치 정보를 저장한다. LLDS(Long-term Location Data Subsystem)는 이동 객체들의 과거 움직임을 저장하는 서브시스템으로, LDP(Long-term Data Processor)와 LDP 노드들의 균형을 유지하기 위한 조정기(Coordinator)로 구성되어 있다. LLDS내의 각 LDP들은 하나의 데이타 프로세서에 대응되며, 상용 데이타베이스 엔진과 연동하여 이동객체의 과거 위치 정보를 저장한다.



<그림 4> GALIS 구조

GALIS는 TMO(Time-Triggered Message Triggered)라는 분산 스킴을 적용한다[4]. TMO를 적용한 GALIS 구조에서 질의 처리 부분은 QueryProc_TMO 이다. QueryProc_TMO는 클라이언트로부터 요청되는 쿼리를 처리하기 위한 TMO이다. 클라이언트로부터 쿼리를 받은 마스터 SDP의 QueryProc_TMO는 필요에 따라 Query_PickupDecompose함수를 사용하여 해당 쿼리를 서브 쿼리들로 분해한 뒤, 워커 SDP들로 분해된 서브 쿼리들을 전송하고, 그 결과를 Resemble-Results함수로 종합하여 클라이언트로 전송 한다

3.1 시공간 모델링

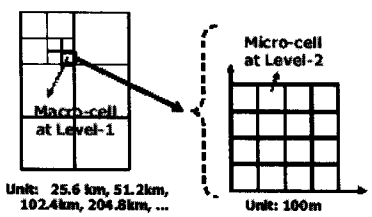


<그림 5> GALIS에서 시공간 모델[6]

일반적으로 실 세계 객체는 움직이는 객체와 움직이지 않는 객체로 구분된다. 예를 들어, 움직이는 객체로는 핸드폰 가입자, 택시, 트럭, 구급차, 열차, 비행기 등이 있다. 움직이지 않는 객체로는 호텔, 병원, 식당 등이 있다. 움직이는 객체의 일반적인 속성으로는 객체 ID(OID), 특정 점(point)으로 표현되는 현재 위치, 시간 순서에 따른 과거 궤적, 현재 시간, 움직인 속도, 움직인 방향등이 있다. 본 논문에서 사용하는 시공간 분할 모델의 구조는 그림 에서와 같이 공간적으로 n개의 2차원 공간으로 분할되며 시간적으로는 p개의 1차원 시간 영역을 가진다.

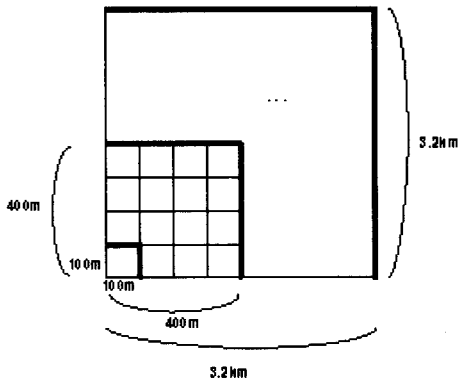
3.2 2단계 공간 인덱싱 기법

본 논문에서 공간 인덱싱 구조로 사용하는 비 균등 2단계 고정 그리드 구조는 이동 객체의 이동정보를 저장 시에 트리 기반 인덱싱 에서의 문제점을 해결하기 위해 고정 그리드를 기반으로 한 2단계의 인덱싱 구조이다. 그림 6은 이동 객체의 위치 정보를 인덱싱 하기 위한 비균등 2단계 고정 그리드 구조를 나타낸다. 1단계는 전체 지역을 담당하는 최 상위 단계의 비 균등 그리드 구조이고 2단계는 균일한 크기로 분할되는 균등 그리드 구조이다. 1단계에서는 인구 밀도나 기타 정보에 따라서 지리적인 지역으로 분할되며 기본적으로 25.6 km를 단위로 해서 사각형 형태로 분할된다. 소위 이러한 사각형을 매크로 셀(macro-cell)이라 하며, 담당하는 지역에 따라 그 크기가 달라지게 된다. 1단계의 각 셀을 한 컴퓨터가 담당하는 것으로 가정하였는데 각 셀에 객체수가 일정 양 이상으로 증가하면 셀이 분할되어 각각을 다른 컴퓨터가 담당하게 된다. 1단계의 각 셀은 담당하는 지역의 객체 수에 따라 분할, 합병되어 각각 컴퓨터를 할당함으로써 대량의 데이타가 집중되어 부하가 집중되는 현상을 막을 수 있다.



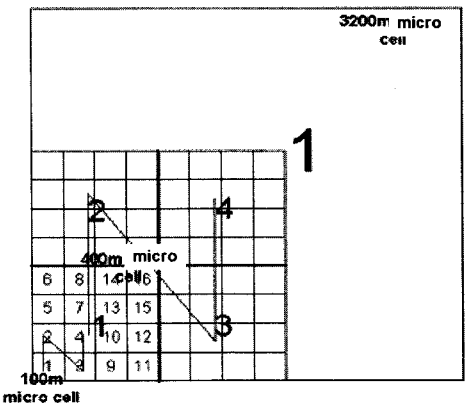
<그림 6> 위치 정보의 인덱싱을 위한 비균등 2단계 그리드 구조[6]

2단계에서는 기본적으로 각 영역이 균일한 형태인 100m X 100m 크기로 분할되며 마이크로 셀(micro-cell)이라 한다. 본 논문에서는 2단계에서의 균등 그리드 구조를 고정 그리드 구조라 명명하고 시간을 고려했을 경우의 질의 처리 방안을 기술한다.



<그림 7> 2단계에서의 셀 종류

<그림 7>은 2단계에서의 셀 종류를 나타내는 그림이다. 기본적으로 2단계는 100m*100m의 셀인 마이크로 셀로 이루어지며, 본 논문에서 적용한 시간 인덱싱 방안인 Zone 개념을 제공하기 위해 2개의 마이크로 셀을 추가로 사용한다. 400m*400m 크기의 매크로 셀은 Zone 2에서 공간적으로 데이터를 걸러내는 기준이 되는 셀이며 3200m*3200m 크기의 매크로 셀은 Zone 3, 4에서 공간적으로 데이터를 필터링하는 역할을 한다.

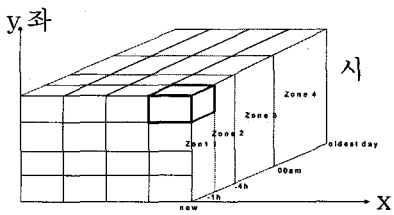


<그림 8> Z-순서를 이용한 셀 번호 채우기

<그림 8>은 2단계에서 Z-순서를 이용해 각 셀에 공간적으로 번호를 채우는 그림이다. 셀 번호는 각 객체의 위치 정보 저장 시에 100m 마이크로 셀, 400m 마이크로 셀, 3200m 마이크로 셀의 번호를 구해서 저장한다. 2단계 고정 그리드 구조에서 100m 마이크로 셀의 경우 100m를 벗어나 움직이는 경우에만 데이타베이스에 반영되며 400m와 3200m 마이크로 셀의 경우도 해당 셀을 벗어나는 경우에만 데이타베이스에 반영되므로 위치정보를 반영하기 위한 갱신 연산이 감소한다.

3.3 Time Zone 기법을 이용한 시간 인덱싱

이동 객체의 위치정보는 계속적으로 변화하므로 샘플링 간격을 길게 한다고 해도 대량의 정보가 된다. 대량의 정보를 하나의 테이블에 저장할 경우 데이타 검색 시간이 기하급수적으로 증가하게 된다. 따라서, 본 논문에서는 위에서 설명한 위치 정보 저장 테이블을 Zone 이라는 시간의 범주에 따라서 시간대 별로 다른 테이블에 정보를 저장한다.

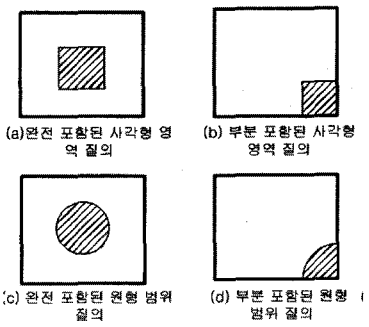


<그림 9> 4개의 Time Zone 개념이 적용된 시공간

<그림 9>는 4개의 Time Zone의 개념이 적용된 시공간 데이타 모델이며, X, Y축으로 이루어진 각 면은 Z-순서를 이용한 고정 그리드이며 Z축은 시간에 따른 변화를 나타낸다. Z축의 Zone1은 100m 이상 움직였으며 현재 시간으로부터 1시간 전의 위치정보를 저장하고 있으며, Zone2는 400m 이상을 움직였으며 1시간에서 4시간사이의 위치 정보를 저장한다. Zone 3는 3.2km 이상을 움직이고 4시간 이후부터 그날의 시작까지의 위치 정보를 저장하며, Zone4는 이전날짜의 데이타로써 백업의 역할을 한다. 해당 Zone은 각각 해당 테이블에 저장되며 검색 시간대에 따라 해당 Zone의 테이블을 검색하면 된다.

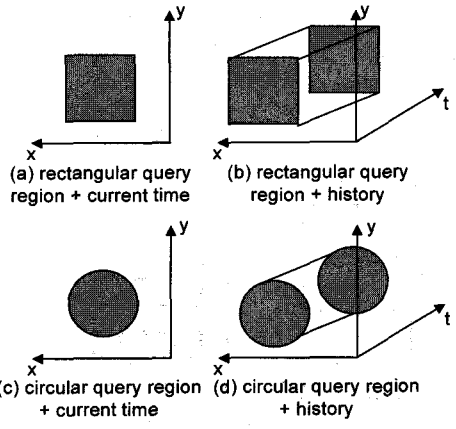
4. 스케쥴링 기법을 이용한 시 공간 질의 처리

4.1 시공간 질의 유형



<그림 10> 다중 노드에서 범위 질의 유형

<그림 10>은 다중 노드에서 원형 질의나 사각형 질의 시에 발생 할 수 있는 여러 가지 질의 형태이다. (a)는 해당 노드에 사각형 질의가 완전히 포함되어 있는 상태이다. (b)는 사각형 질의 일부가 해당 노드에 포함되어 있는 경우이다. (b)의 경우에 부분적으로 포함된 사각형 질의와 노드 좌표계 간의 겹치는 좌측 하단 좌표와 우측 상단 좌표를 구하는 작업이 필요하다.v 사각형 질의 시에는 (a), (b) 모두 사각형 필터링(rectangular filtering)을 적용할 수 있다. (c)는 원형 질의가 해당 노드에 완전히 포함된 경우이다. (d)의 경우에는 원형 질의가 해당 노드에 일부 포함된 상태이며 포함된 모든 객체와 질의 중심점 간의 거리계산이 필요하다.

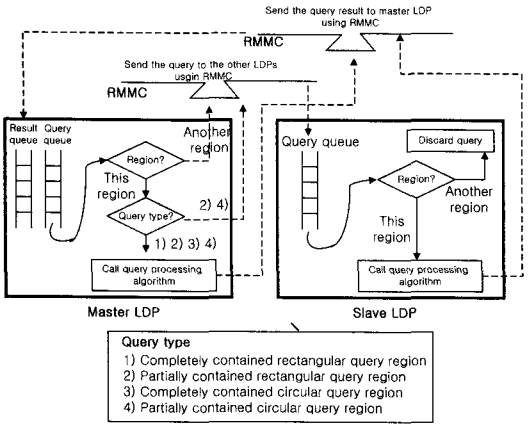


<그림 11> 시공간 질의 유형

<그림 11>은 시공간 질의 유형을 나타낸다. 그림(a)와 (b)는 사각형 질의에 시간을 추가한 질의 유형이다. (a)는 현재 사각형 범위 내에 있는 객체를 검색하는 질의이다. (b)는 특정 시간 범주를 가지는 사각형 범주 질의 이다.

<그림 11>의(c)와 (d)는 원형 질의에 시간을 추가한 질의 유형이다. (c)는 현재 원형 범위 내에 있는 객체를 검색하는 질의이다. (d)는 특정 시간 범주를 가지는 원형 범주 질의이다. 그림 11에서와 같이 시공간 질의 유형의 경우 공간적인 검색과 시간적인 검색이 모두 이루어 져야 한다.

4.2 GALIS 구조에서 시공간 질의 처리 흐름



<그림 12> LLDS 질의 처리 흐름

LLDS는 하나의 마스터 LDP와 여러개의 워커 LDP들로 구성된다. <그림 12>는 GALIS 구조에서 LLDS에 질의가 들어온 이후에 질의가 처리 되는 과정이다. 질의는 마스터 LDP의 질의 큐에 질의가 계속적으로 쌓이고 있다. 마스터 LDP에서는 큐에서 질의를 꺼내어 현재 영역이면 현재 노드에서 처리하고, 아니면 RMMC를 통해 다른 노드로 질의를 전송한다. 마스터 노드에 포함되는 질의 인 경우 질의 형태에 따라 해당되는 질의 처리 알고리즘을 호출한다. 각 질의 중에서 현재 마스터 노드에 부분적으로 포함되어 있는 질의는 RMMC를 통해 다른 노드로 전송한다.

워커 노드에 포함되는 질의인 경우 각 큐에서 질의 꺼내서 현재 영역에 포함되는지 검사한다. 현재 영역에 포함되면 질의 형태에 따라 해당되는 질의 처리 알고리즘을 호출한다. 현재 영역에 포함되지 않는 질의인 경우 아무런 처리없이 버린다. 질의 처리 알고리즘을 수행하여 질의 처리가 끝나면 RMMC를 통해 질의 결과 큐로 전송된다. 질의 결과 큐에서는 질의 ID 별로 질의 결과를 통합하여 사용자에게 반환한다.

4.3 시공간 질의 스케쥴링

시공간 질의 스케쥴링은 질의 큐에 입력된 일정 질의를 특정 시간적, 공간적, 다른 처리기와의 관련성을 고려하여 처리 순서를 재조정함으로써 질의 처리율을 높이는 기법이다. 본 논문에서는 질의 처리 순서를 결정하는 관련성으로 위치 관련성, 시간 관련성, 노드 관련성 3가지를 고려한다. 각 관련성에 대한 가중치는 공간적으로 인접한 지역에 대해서 질의가 많은 경우 위치 관련성의 가중치를 높게 하고, 시간의 범위가 유사한 질의가 많은 경우 시간 관련성의 가중치를 높게 한다. 분산된 노드에 대해서 질의가 분산되어 질의가 입력되는 경우 노드 관련성의 비중을 높일 수 있다.

4.3.1 셀 번호를 이용한 위치 관련성

일반적으로 질의 집중 지역에 대해 수행된 질의들은 서로 공간적으로 겹쳐 있는 경우가 많고, 겹쳐진 지역 내에 존재하는 셀을 사용할 확률이 크다. 질의 간 위치 관련성은 영역 질의 간의 공간 관련성을 나타내는 기준이 된다. 질의 간 위치 관련성은 다음과 같다.

[정의 1] 질의 간 위치 관련성

LR = COUNT(Overlap_Cell_No(Qprevious , Qcurrent))

LR(Location Relationship) = 질의 Q의 위치 관련성

Overlap_Cell_No(Q1, Q2) = 질의 Q1과 Q2가 겹치는 셀의 번호

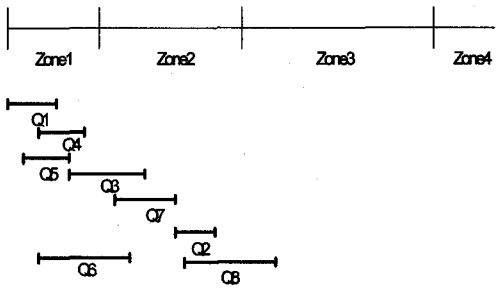
Qprevious = 위치 관련성 비교 대상 기준 질의

Qcurrunt = 현재 대기 중인 질의

위의 식에서 보이는 바와 같이 질의 간 위치 관련성은 대상 질의에 대해 n개의 영역 질의 들이 서로 겹치는 셀의 개수 있다. 질의 간 위치 관련성은 질의들이 겹치 셀의 개수에 비례하게 된다. 질의 간 위치 관련성이 크다는 뜻은 해당 질의가 서로 인접한다는 뜻으로 서로 같은 같은 데이타를 공유 할 확률이 높다는 것이다. 위치 관련성이 높은 질의들을 처리하게 되면

근접한 지역에 대해 집중되는 영역 질의를 같이 처리하게 되므로 데이타를 찾을때 필요한 탐색의 중복을 최소화 할 수 있다.

4.3.2 Time Zone 개념을 이용한 시간 관련성



<그림 13> Zone개념 에서 질의 간 시간 관련성

질의 간 시간 관련성 이란 주어진 질의의 시간 범위간의 겹치는 정도를 의미한다. 공간 관련성과 마찬가지로 시간 범위가 주어진 궤적 질의의 경우 서로 시간적으로 겹쳐 있는 경우가 많이 발생할 수 있으며 시간적으로 겹쳐진 곳의 데이타를 공유할 확률이 높다. <그림 13>에서 각 Zone은 하나의 테이블을 의미하므로 같은 Zone내에 있는 질의들을 인접하여 처리하는 것이 좋고 기준 질의에 대해서 시간적으로 겹치는 정도가 높은 질의를 우서 처리하는 것이 중복 탐색을 최소화 할 수 있다. Zone 개념을 이용한 질의 간 시간 관련성은 다음과 같다.

[정의 2] 질의 간 시간 관련성
TR = Overlap_Time(Qprevious , Qcurrent)
LR(Time Relationship) = 질의 Q의 위치 관련성
Overlap_Time(Q1, Q2) = 질의 Q1과 Q2가 겹치는 시간
Qprevious = 시간 관련성 비교 대상 기준 질의
Qcurrunt = 현재 대기 중인 질의

4.3.3 노드 관련성과 가중치 조절

질의 스케쥴링을 위해서는 질의 처리 큐에 대기하고 있는 질의들의 우선 순위를 결정해야 한다. 본 논문에서는 공간적 우선순위로서 4.3.2의 위치 관련성을 사용하고, 시간적 우선순위로 4.3.3의 시간 관련성을 이용한다.
본 논문에서 가정하는 질의는 여러 노드에 대해서 겹쳐서 수행되는 범위 질의를 사용하므로 노드간의 관련성도 고려하여야 한다. 노드 관련성에 대한 정의는 다음과 같다.

[정의 3] 질의 간 노드 관련성
NR = COUNT(Overlap_Node_No(Qcurrent))
NR(Node Relationship) = 질의 Q의 노드 관련성
Overlap_Node_No(Q1) = 질의 Q1와 겹치는 노드 번호
Qcurrunt = 현재 대기 중인 질의

다음은 각 질의에 대해서 위치 관련성과 시간 관련성의 비율을 조정하는 식이다.

[정의 4] 각 관련성간의 가중치 조절후의 질의 관련성
QR = LR*WL + TR*WT + NR*WN
QR(Query Relationship) = 질의 Q의 관련성
WL = 위치 관련성에 대한 가중치,
WT = 시간 관련성에 대한 가중치,
WN = 노드 관련성에 대한 가중치

4.3.4 질의 처리 스케쥴링을 위한 자료구조와 알고리즘

query_queue

RANGE spatial_range	// 질의 공간 범위
INT time_range	// 질의 시간 범위
FLOAT priority	// 질의 우선 순위
DATETIME time	// 질의 입력 시간
void clear()	// 질의큐 초기화
void insert()	// 질의 입력
void delete()	// 질의 삭제

<그림 14> 질의 큐 자료구조

일정 시간 안에 입력된 질의를 처리하기 위해서는 질의를 저장하여 처리할 수 있는 자료구조가 필요하다. 그림 14는 큐 형태의 질의 저장 자료구조를 나타낸다. 일반적인 큐는 FIFO(First-In First-Out)형태로 구성되는데 비해 질의 처리 큐는 큐 안의 질의들 중 가장 높은 우선 순위를 가진 질의를 먼저 출력하는 우

선순위 큐 이다. 질의 큐 자료구조는 질의의 공간적 범위와 시간 범위, 그리고 우선순위, 입력시간으로 구성된다. 질의 입력 시간은 질의가 입력된 후 우선순위가 낮아 계속 연기되는 기아(starvation)현상을 해결하기 위해서 사용한다. 우선순위가 매우 낮아 계속 적으로 연기되는 질의의 경우 질의 입력 시간을 이용하여 노화(aging) 기법으로 해결한다.

질의 우선순위는 질의 큐 내의 질의들에 부여되는 우선순위로써 질의를 큐에서 꺼낼 때 사용한다. 질의 큐 내의 질의의 처리를 위한 메소드로는 질의큐 질의를 초기화, 입력, 삭제하는 메소드가 있다.

```
• 질의 스케쥴링
query_scheduling (query_queue[], prev_query[])
  //prev_query[]는 이전에 수행되었던 질의들 중 기존질의 인자는 1
  FOR EACH(cur_query in query_queue[])
    FOR EACH(prev_query in prev_query[])
        location_relationship[] = COUNT(overlap_cell_no(prev_query, cur_query)) //위치관련성
        time_relationship[] = overlap_time(prev_query, cur_query)//시간관련성
        node_relationship[] = COUNT(overlap_node_no(cur_query)) //노드관련성
    END FOREACH
  END FOREACH
  relationship[]= adjust_weight(location_relationship[], time_relationship[], node_relationship[]
  set_priority(query_queue[], relationship[]) //가중치 조절후 우선순위값 결정
END query_scheduling
• 질의 큐에서 최고 우선순위의 질의를 선택
get_next_query(query_queue[])
  query = max_priority(query_queue[])
  call procedure scheduling_search(query)
END get_next_query
```

<그림 15> 질의 스케쥴링 및 질의 선택 알고리즘

<그림 15>는 질의 스케쥴링 알고리즘과 우선순위가 부여된 질의 큐에서 질의를 선택하는 알고리즘이다. 질의 스케쥴링 알고리즘은 질의 큐에 있는 질의들의 우선순위를 결정하는 알고리즘이다.

질의 큐 내에 있는 질의들의 우선순위를 결정하기 위해서는 기준질의와 각 질의의 우선 위치 관련성과 시간 관련성 그리고 노드 관련성을 구해야 한다. 각 관련성을 구한 후 가중치 조절을 해서 통합 관련성을 구한다.

구한 통합 관련성 값을 이용해서 질의 큐 배열의 우선 순위를 조정한다.

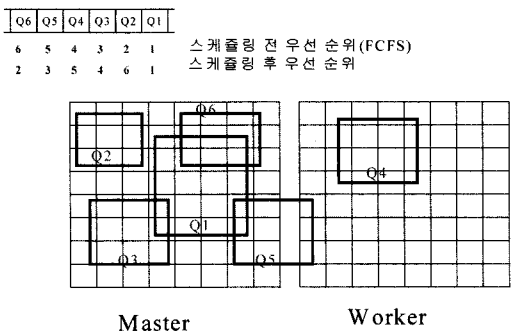
질의 큐 내에 있는 질의의 선택을 위해서는 스케쥴링을 통해 구한 우선순위 값이 가장 높은 값을 가장 먼저 선택한다. 만약 우선순위 값이 같을 경우에는 FCFS를 적용한다.

```
• 질의 검색
  scheduling_search( query )
  query_result[] = search result of prev_query[] //기준 질의의 결과를 query_result에 삽입
  IF region of current region is overlapped for prev_query[]
    search for query_result[]
  ELSE
    search for database table
  END IF
END scheduling_search
```

<그림 16> 스케쥴링을 적용한 질의 검색 알고리즘

<그림 16>은 스케쥴링 기법을 적용했을 때 질의를 수행하는 알고리즘이다. 질의 큐에 꺼내진 질의에 대해서 기준 질의와 겹치는 영역에 대해서는 queyr_result[]에서 질의를 하고 겹치지 않는 영역에 대해서는 데이터베이스에서 검색을 한다. 기준 질의에 대해서 겹치는 영역에 대해서는 데이터베이스 재 검색의 오버헤드가 없어서 불필요한 중복 탐색 시간을 줄일 수 있다.

4.3.5 질의 스케쥴링 예



(a) 2개 노드에서의 질의 예

$$QR = LR * W_L + TR * W_T + NR * W_N$$
$$= LR * 2/3 + TR * 0 + NR * 1/3$$

Q2 = 0 * 2/3 + 1 * 1/3 = 0.33
Q3 = 2 * 2/3 + 1 * 1/3 = 1.67
Q4 = 0 * 2/3 + 2 * 1/3 = 0.67
Q5 = 2 * 2/3 + 2 * 1/3 = 2
Q6 = 6 * 2/3 + 1 * 1/3 = 4.33

(b) 질의 관련성 계산

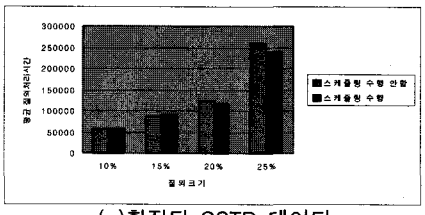
<그림 17> 질의 스케쥴링 예

<그림 17>은 본 논문에서 제안한 질의 스케쥴링 기법의 예를 보이고 있다. 그림 17의 (a)에서와 같이 master와 worker 2개의 노드가 있으며 master 노드가 질의 스케쥴링을 전담하는 예이다. 질의 큐에 질의가 6개가 있을 때 기준질의는 첫 번째 질의 Q1으로 가정하고 스케쥴링을 수행한다. 위의 예에서는 질의 예제의 간략화를 위해 시간 관련성은 고려하지 않았으며 위의 예에서 노드 분산은 많지 않으며 공간적으로 관련성이 높은 질의들이 많으므로 공간관련성의 가중치를 높게 하였다.

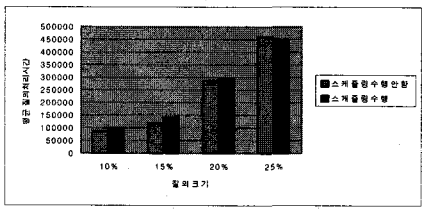
기준 질의 Q1에 대해서 Q2~Q6의 위치 관련성, 노드 관련성을 구한다. 위의 예에서는 시간 관련성은 고려하지 않는다. 구한 각 관련성에 가중치를 부여하여 통합 관련성을 구한다. 통합 관련성의 값을 이용하여 우선순위를 부여하면 위치 관련성이 높은 Q6이 두 번째가 된다. Q3과 Q5는 위치 관련성은 같지만 Q5가 노드 관련성 값이 높으므로 Q5 다음으로 Q3이 실행된다. 기준질의 Q1에 대해서 위치 관련성과 노드 관련성이 낮아 제일 나중에 수행되게 된다.

5. 성능

본 논문에서 제안한 스케쥴링 기법의 유용함을 보이기 위해 실험 하였다. 스케쥴링 기법을 이용한 성능 평가를 위해서 질의 스케쥴링 알고리즘의 적용 유무에 따른 성능 차이를 비교한다. 성능 평가를 위해 LDP master와 LDP worker 두 개의 노드를 가정하고 실험 하였다. 실험을 위한 전체 영역은 25.6km*25.6km 를 가정하였고 그림 19에서는 이동객체의 수는 2000개 이고, 질의의 수는 100개를 질의 크기별로 생성하여 실험하였고 그림 20에서는 질의 범위를 20%로 고정시키고 질의 수와 객체 수를 변화 시켜가며 실험하였다. 이동객체의 위치정보를 발생시키는 시뮬레이션 데이타로는 확장된 GSTD와 사용자 패턴을 적용한 시뮬레이터의 데이타를 이용하였다.



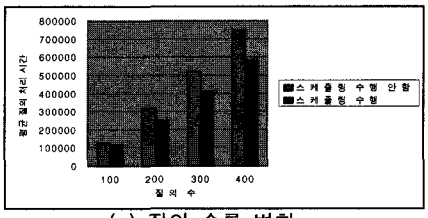
(a)확장된 GSTD 데이타



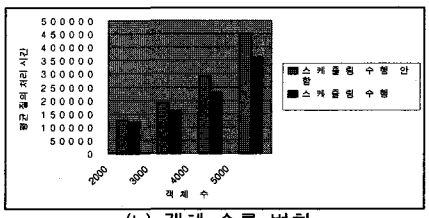
(b) 사용자 패턴을 적용한 데이터

<그림 18> 질의 범위에 따른 질의 처리 시간 비교

<그림 18>은 확장된 GSTD 데이타와 사용자 패턴 데이타를 이용해 질의 범위를 변화 시켜가며 스케쥴링 유무에 따른 질의 처리시간을 비교한 그림이다. 그림에서 (a) 와 (b) 모두 10%와 15%일 경우 질의 스케쥴링을 수행했을 경우가 질의 시간이 오히려 시간이 조금씩 증가했는데 그 이유는 질의 영역이 작아서 질의간의 위치 관련성이 적어서 스케쥴링에 의한 오버헤드가 더 크기 때문이다. 질의 영역이 큰 경우 질의간의 위치 관련성이 커져서 각 질의간에 공유하는 데이타가 많아져서 질의 수행 시간이 줄어드는 것을 알 수 있다. (a)와 (b)의 20%에서 결과가 다른 이유는 확장된 GSTD와 사용자 패턴을 적용한 데이타의 분포가 다르기 때문이다.



(a) 질의 수를 변화



(b) 객체 수를 변화

<그림 19> 질의 수와 객체 수에 따른 질의 처리 시간 비교

<그림 19>는 확장된 GSTD 데이타를 이용하여 질의 수와 객체수를 변화 시켜 가며 실험한 결과 이다. (a)와 (b) 모두에서 스케쥴링을 수행하였을 경우 평균 질의 처리 시간이 감소하는 것을 알 수 있다. 그림 18와 19의 결과를 비교해 보면 질의 객체의 분포 보다는 질의 수와 질의 대상 객체의 수에 영향을 받는 것을 볼 수 있다. 즉, 질의 수와 객체수가 많아질수록 스케쥴링의 효과가 커지는 것을 알 수있으며 대량의 객체 정보와 대량의 질의에 효율적인것을 알 수 있다.

6. 결 론

이동 객체의 현재 위치와 과거 위치를 기반으로 하는 위치 기반 서비스가 점차 증가하고 있다. 이러한 위치 기반 서비스의 저장 및 인덱싱의 대상이 되는 이동 객체는 시간에 따라서 위치가 계속적으로 변화하며 시간이 지남에 따라 그 양은 기하급수적으로 증가하게 된다. 또한, 이동 객체를 핸드폰 가입자라고 했을 때 이동은 불규칙적이며 특정 노드에 집중되는 현상이 발생할 수 있다. 이 경우 부하를 분산 시킬 수 있는 방안이 있어야 하며 누적되는 데이터의 양을 줄일 수 있는 방법이 있어야한다. 또한 이동 객체의 경우 현재 위치가 계속 변화 하므로 그 변화에 따라서 전체 인덱스 구조의 변화를 최소화 할 수 있는 방안이 있어야 한다.

본 논문에서는 이동 통신 서비스에서 계속적으로 이동하는 고객의 위치정보를 저장 및 검색하기 위해 비 균등 2단계 고정 그리드 인덱싱 방법을 사용하여 처리했다. 기존의 R-트리 계열의 인덱싱 방법은 이동 객체의 움직임에 따라 갱신 연산이 이루어지므로 위치 정보를 인덱싱 하기에는 부적합 하다고 볼 수 있다.

본 논문에서 적용한 GALIS 시스템의 비 균등 2단계 고정 그리드 인덱싱에서 1단계는 담당하는 지역의 객체 수에 따라 동적으로 변하는 단계이며, 2단계는 특정 지역을 균등 분할하여 사용했다. 따라서 비 균등 2단계 고정 그리드 인덱싱을 사용하면 부하를 분산 시킬 수 있는 효과가 있다. Time Zone 기법을 사용하면 과거로 갈수록 데이터의 양이 줄어들어 검색 시간을 감소시킬 수 있다. 또한 시공간 영역 질의 들은 동일한 영역을 참조하는 경우가 많다. 이러한 관련성이 높은 질의들은 우선순위를 부여하여 연속적으로 처리하는 것이 효율적이다.

본 논문에서는 질의간의 관련성을 공간, 시간, 노드로 정의 하고 이들 간의 관련성을 이용해 우선 순위를 부여하여 동일 데이타의 중복 탐색을 최소하 하는 스케쥴링 기법을 제안 하였다. 본 논문에서 제안한 질의 스케쥴링 기법의 유용함을 보이기 위해 성능평가 하였다. 성능 평가 결과 질의 영역이 큰 경우 질의 스케쥴링을 이용하면 효율적이라는 것을 알 수 있었다. 향후 연구 과제로는 다양한 형태의 시공간 질의에 대해서 스케쥴링을 수행할 수 있도록 스케쥴링 기법을 확장하고 실험 평가가 이루어져야 한다.

참고문헌

[1] J.A.Orenstein and T.H. Merrett. A class of data structures for associative searching, Proc. of SIGACT-SIGMOD, pages 181-190, April 1984

[2] 박원순, 전세길, 나연묵, "대용량 이동 객체의 위치 정보 인덱싱", 정보과학회 추계학술발표논문집, Vol. 29, No. 2, 2002년 10월,pp 49-51.

[3] 전세길, 나연묵, "고정그리드 인덱싱에서 VP 필터링을 이용한 범위 질의 처리", 정보처리학회 춘계학술발표논문집, 제10권 1호, 2003년5월, pp 1531-1534

[4] Y.Nah, M.H.Kim, T.Wang, K.H.Kim, Y.K.Yang, "TMO-structured Cluster-based Real-Time Management of Location Data on Massive Volume of Moving Items," STFES 2003, Hakodate, Japan.

[5] M.H.Kim, T.Wang, K.H.Kim, Y.Nah, Y.K.Yang, "Distributed Adaptive Architecture for Managing Very Large Volume of Moving Items," IDPT 2003, Beijing, China.

[6] 나연묵, K. H. Kim, 왕태형, 김문회, 이종훈, 양영규, "GALIS: LBS 시스템의 클러스터 기반 신축성소유 아키텍처", 데이터베이스연구, 제18권 4호, 2002년 12월, pp 33-47

[7] Segil Jeon, Yunmook Nah, "Range Query Processing for Distributed Real-time Moving Object Databases", APIS 2004, Istanbul, Turkey.

[8] Saltenis, S., et al, Jensen, E., Leutenegger, S., Lopez, M., "Indexing the Positions of Continuously Moving Objects" Proc. ACM SIGMOD, 2000.

[9] Theodoridis, Y., Vazigannis, M., and Sellis, T., "Spatio-Temporal Indexing for Large Multimedia Applications", Proc. 3rd IEEE Conf. on Multimedia Computing and Systems(ICMCS), 1996

[10] J. Nievergelt, H. Hinterberger, and

K.C.Sevcik. The grid file: an adaptable, symmetric multikey file structure. ACM TODS, 9(1): 38-71, March 1984.

[11] Xu, X., Han, J., and Lu, W., "RT-tree: An Improved R-tree Index Structure for Spatiotemporal Databases", Proc. 4th Int'l Symp. on Spatial Data Handling(SDH), 1990.

[12] Jeffrey K. Ujlmann, "Satisfying General Proximity/Similarity Queries with Metric Trees, " Information Processing Letters, Vol. 40, pages 175-179, 1991.

[13] Tolga Bozkaya and Meral Ozsoyoglu, "Distance-Based Indexing for High-Dimensional Metric Spaces," Proceedings of the ACM SIGMOD Conference, pages 357-368, 1997.

[14] P.N.Yiannilos, "Data Structures and Algorithms for Nearest Neighbor search in General Metric Spaces," ACM-SIAM Symposium on Discrete Algorithms, 1993, pages 311-321.

[15] Philippe Rigaux, Michel, Scholl, Agnes Voisard, *Spatial Databases with Application to GIS*, Morgan Kaufmann.

[16] Yufei Tao, Dimitris Papadias, "Time Parameterized Queries in Spatio-Temporal Databases," ACM SIGMOD 2002.

[17] Terry, D., Goldberg, D., Nichols, D., Oki, B.,"Continuous Queries over Append-only Databases," ACM SIGMOD, 1992.

[18] Chen, J., DeWitt, D.J., Tian, F., Wang, Y,"NiagaraCQ: A Scalable Continuous Query System for Internet Databases," ACM SIGDMOD, 2000.

[19] 박영민, 전봉기, 서영덕, 홍봉희, " 디클러스터링된 공간 데이타베이스 에서의 다중 공간 질의 처리", 한국정보과학회 '99 가을 학술발표논문집, 제26권 2호, pp.314-316, 1999.

[20] 고영균, 나연묵, "위치 기반 서비스를 위한 이동 객체 데이타베이스 구현", 석사학위논문, 2004

[21] Yannis Theodoridis, Jefferson R. O. Silva, Mario A. Naschimento, "On the Generation of Spatiotemporal Datasets", CHOROCHRONOS Technical Report CH-99-01

전세길

1998년 단국대학교 컴퓨터공학과 (공학사)

2000년 단국대학교 대학원 컴퓨터공학과(공학석사)

2004년 단국대학교 대학원 전자컴퓨터공학과 (공학박사)

관심분야: 데이타베이스, 멀티미디어, 시공간데이타베이스

황재일

1990년 단국대학교 전자공학과 (공학사)

1999년 단국대학교 산업대학원 정보처리학과(공학석사)

2000년 ~ 현재 단국대학교 전자 컴퓨터공학과 박사과정

관심분야: 데이타베이스, 멀티미디어, 시공간데이타베이스

나연묵

1993년 ~ 현재 단국대학교 전기전자 컴퓨터공학부 부교수

1996년 ~ 현재 한국정보과학회 데이타베이스연구회 운영위원

2000년 ~ 현재 한국정보과학회 논문지 편집위원

관심분야: 데이타베이스, 객체지향 데이타베이스, 멀티미디어 시스템, 시공간 데이타베이스