대용량 이동 객체 위치 데이타 관리 시스템의 개발†

Development of a Location Data Management System for Mass Moving Objects

김동오*, 주성완*, 장인성**, 한기준*

Dong-Oh Kim, Sung-Wan Ju, In-Sung Jang, Ki-Joon Han

요 약 최근 이동 객체의 위치 데이타를 이용하기 위한 무선 측위 기술과 모바일 컴퓨팅 기술이 급속도로 발전하였다. 또한, 이동 객체의 위치 데이타를 활용하는 위치 기반 서비스에 대한 수요가 급증하고 있으며, 이러한 서비스를 지원하기 위해서는 이동 객체의 위치 데이타를 효과적으로 저장할 수 있는 시스템이 필요하다. 이러한 시스템은 이동 객체의 수가 많고 위치 획득 간격이 짧을수록 위치 데이타가 급격히 늘어나기 때문에 대용량의 위치 데이타 처리가 가능해야 하며, 위치 기반 서비스를 위한 다양한 시공간 질의를 지원해야 하고, 또한 이동 객체의 불확실성 문제를 해결할 수 있어야 한다. 따라서, 본 논문에서는 이동 객체의 위치 데이타를 효율적으로 관리하기 위한 해쉬 기법, 클러스터링 기법, 시간 질의 검색 기법을 제시하였다. 그리고, 대용량의 이동 객체 위치 데이타를 효과적으로 저장하고 검색할 수 있으며, 시공간 질의 기능과 불확실한 과거 위치 데이타 처리 기능을 제공하는 디스크 기반의 대용량 이동 객체 위치 데이타 관리 시스템을 개발하였다. 본 시스템을 SQL-Server과 성능 비교한 결과 이동 객체 저장 성능은 약 5% 증가하였으며, 이동 객체 검색 성능은 약 300% 증가하였다.

Abstract Recently, the wireless positioning techniques and mobile computing techniques were developed with rapidly to use location data of moving objects. Also, the demand for LBS(Location Based Services) which useslocation data of moving objects is increasing rapidly. In order to support various LBS, a system that can store and retrieve location data of moving objects efficiently is required necessarily. The more the number of moving objects is numerous and the more periodical sampling of locations is frequent, the more location data of moving objects become very large. Hence the systemshould be able to efficiently manage mass location data, support various spatio-temporal queries for LBS, and solve the uncertainty problem of moving objects.

Therefore, in this paper, we presented a hash technique, a clustering technique and a trajectory search technique to manage location data of moving objects efficiently. And, we have developed a Mass Moving Object Location Data Management System, which is a disk-based system, that can store and retrieve location data of mass moving objects efficiently and supportthe query for spatio-temporal data and the past location data with uncertainty. By analying the performance of the Mass Moving Object Locations Management system and the SQL-Server, we can find that the performance of our system for storing and retrieving location data of moving objects was about 5% and 300% better than the SQL-Server, repectively.

주요어 : 이동 객체 데이타베이스, 위치 데이타 관리, 위치 기반 서비스

Keywords : Moving Object Database(MODB), Location Data Management, Location Based Services(LBS)

1. 서 론

최근 핸드폰과 같은 모바일 장치들을 활용하는 무선 인터넷 사용자가 수가 급격히 증가하고, 이러한 이동 객체(Moving Objects)의 위치 데이타를 이용하기 위한 무선 측위 기술과 모바일 컴퓨팅 기술이 급속도

† 본 연구는 한국전자통신연구원의 개방형 LBS 핵심 기술 개발과제의 연구비 지원에 의해 수행되었음.

* 건국대학교 컴퓨터공학과 {dokim, swju, kjhan}@db.konkuk.ac.kr

** 한국전자통신연구원 e4dol2@etri.re.kr

로 발전하였다. 또한, 이동 객체의 위치 데이타를 활용한 위치 추적 서비스, 교통 정보 서비스, 모바일 광고 서비스, 긴급 구조 서비스 등과 같은 위치 기반 서비스(LBS: Location Based Services)에 대한 수요가 급증하고 있다[1][2].

다양한 위치 기반 서비스를 제공하기 위해서는 이동 객체의 위치 데이타를 신속하게 저장하고 검색할 수 있는 이동 객체 데이타베이스 시스템이 필수적으로 요구된다[3][4][5]. 특히, 이동 객체는 그 수가 많고 위치 획득 간격이 짧을수록 이동 객체의 위치 데이타가 기하급수적으로 늘어나기 때문에 대용량의 위치 데이타를 효율적으로 관리할 수 있어야 한다[6]. 또한, 위치 기반 서비스에서 이동 객체의 위치 데이타를 이용하기 위해 다양한 시공간(Spatio-Temporal) 질의도 지원되어야 한다[7].

기존의 상용 데이타베이스 시스템은 이동 객체를 처리하기 위한 시스템이 아니므로 이동 객체 처리 시 불필요한 트랜잭션 연산으로 인해 저장 및 검색 오버헤드가 발생하게 된다[8]. 또한, 기존의 데이타베이스 시스템은 이동 객체에 대한 다양한 시공간 질의를 지원하지 않기 때문에 SQL을 사용하여 이동 객체 질의를 표현하는 것이 어렵고 질의 처리 성능이 저하되며, 특히 데이타베이스에 저장되지 않은 위치 데이타로 인해 발생되는 이동 객체의 불확실성(Uncertainty) 문제를 처리하지 못한다[9][10][11].

이러한 문제점을 해결하기 위해서 본 논문에서는 대용량 이동 객체의 위치 데이타를 효과적으로 저장하고 검색할 수 있으며, 시공간 질의 기능과 불확실한 과거 위치 데이타 처리 기능을 제공하는 대용량 이동 객체 위치 데이타 관리 시스템을 개발하고 성능 평가를 수행하였다.

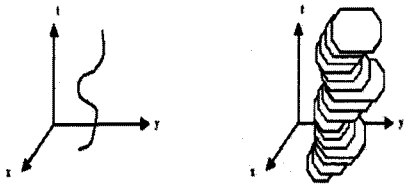
본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 이동 객체와 .NET Remoting에 대해 살펴본다. 제 3장에서는 본 논문에서 사용한 이동 객체 위치 데이타 관리 방법에 대해 기술하고, 제 4장에서는 대용량 이동 객체위치 데이타 관리 시스템의 설계에 대해서 설명한다. 제 5장에서는 대용량 이동 객체 위치 데이타 관리 시스템의 구현 및 성능 평가에 대해 상세히 설명한다. 마지막으로, 제 7장에서는 결론에 대해서 언급한다.

2. 관련 연구

본 장에서는 이동 객체에 대해 살펴보고, 대용량 이동 객체 위치 데이타 관리 시스템에서 클라이언트의 원격 접속을 지원하기 위해 사용된 .NetRemoting에 대해서 알아본다

2.1 LMG-tree의 특징

이동 객체란 시간의 흐름에 따라 연속적으로 위치와 모양이 변하는 시공간 객체를 말하며, 크게 이동점(moving point)과 이동 영역(moving region)으로 나뉜다[12][13]. 이동점은 <그림 1>(a)와 같이 시간에 따라 위치가 변하는 객체로서, 그 예로는 사람, 동물, 자동차, 선박, 비행기 등이 있다. 이동 영역은 <그림 1>(b)와 같이 시간에 따라 위치와 모양이 변하는 객체로서, 그 예로는 태풍, 기온 변화, 암세포 등을 들 수 있다. 본 논문에서 구현한 대용량 이동 객체 위치 데이타 관리 시스템에서는 이동 객체를 이동점으로 제한한다.



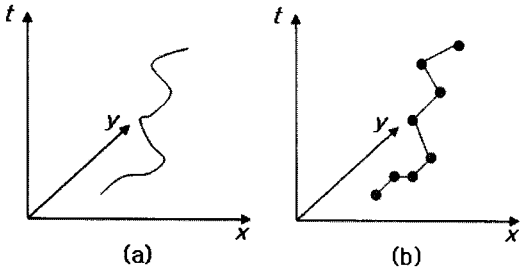
(a) 이동 점 (b) 이동 영역

<그림 1> 이동 점과 이동 영역

2.1.1 이동 객체 데이타 모델

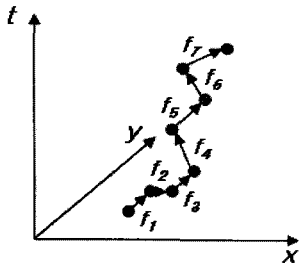
이동 객체를 표현하는 데이타 모델에 관한 기존 연구는 크게 두 가지로 분류된다. 하나는 이동 객체의 과거 위치와 현재 위치를 다루는 데이타 모델로서, 이동 객체에 대한 궤적(Trajectory) 질의가 가능하다[12][14][15]. 이 데이타 모델은 추상적(Abstract) 모델과 이산적(Discrete) 모델로 나뉘는데, 전자는 <그림 2> (a)와 같이 이동 객체의 궤적을 무한한 점들의 집합으로 나타내며, 이를 3차원 공간상의 연속적인 곡선(Curve)으로 표현한다. 반면에, 후자는 <그림 2>(b)와 같이 이동 객체의 궤적을 유한한 점들의 집합으로 나타내며, 이를 3차원 공간상의 선분(PolyLine)으로 표

현한다. 본 논문에서는 이산적 모델과 같이 이동 객체의 궤적을 유한한 점들의 집합으로 나타낸다.



<그림 2> 추상적 모델과 이산적 모델

또 다른 하나는 이동 객체의 현재 위치를 시간에 따라 연속적으로 변하는 동적 속성(Dynamic Attribute)을 사용하여 표현하는 MOST(Moving Object Spatio-Temporal) 데이타 모델로서, 이동 객체의 현재 위치뿐만 아니라 미래 위치까지 예측할 수 있다[16]. 이 데이타 모델은 eventually, always, until, nexttime 등의 시간 연산자를 제공하는 FTL(Future Temporal Logic) 언어를 사용하여 미래 질의를 표현할 수 있도록 하고 있다. <그림 3>은 이동 객체의 현재 위치를 유지 관리하기 위해 시간 함수 f를 사용한 MOST 데이타 모델을 보여준다.



<그림 3> MOST 모델

2.1.2 이동 객체 질의 유형

이동 객체에 대한 질의 유형은 크게 Timestamp 질의, 영역 질의, 궤적 질의, 복합 질의로 나뉜다[17]. Timestamp 질의는 특정 시간에 주어진 영역에 속하는 이동 객체를 검색하는 질의이며, 영역 질의는 주어진 시간 간격 동안에 주어진 영역에 속하는 이동 객체를 검색하는 질의이다. 궤적 질의는 특정 이동 객체의 궤적을 검색하는 질의이다. 그리고 복합 질의는 영역 질의와 궤적 질의가 결합된 질의로서, 특정 시간에 주어진 영역을 지나간 객체의 궤적을 검색하는 질의가 이에 해당한다. 본 논문에서 구현한 대용량 이동 객체 위치 데이타 관리 시스템은 이동 객체에 대한 시공간 질의를 위해서 영역 질의와 궤적 질의를 제공한다.

그밖에 이동 객체에 대한 질의로는 최근접(Nearest-Neighbor) 질의, 위상(Topological) 질의, 항해(Navigation) 질의 등이 있다[18]. 최근접 질의는 특정 시간에 주어진 지점에서 가장 가까운 곳에 있는 객체를 검색하는 질의이며, 위상 질의는 이동 객체의 궤적에 대한 위상 정보를 포함하는 질의이고, 항해 질의는 이동 객체의 속도, 방향과 같은 이동 객체의 궤적 정보로부터 유도될 수 있는 정보를 포함하는 질의이다.

2.2 .NET Remoting

.NET Remoting은 .NET 프레임워크를 통해 다양한 응용 프로그램 도메인, 프로세스, 컴퓨터에 있는 객체들이 서로 통신할 수 있도록 해주는 원격 기술이다[19]. 또한 .NET 프레임워크는 원격 응용 프로그램에서 메시지를 주고받는 통신 채널뿐만 아니라 활성화 및 수명 지원을 포함한 다양한 서비스를제공한다. 본 논문에서 구현한 대용량 이동 객체 위치 데이타 관리 시스템은 .NET Remoting을 사용하여 클라이언트의 원격 접속을 지원하고 있다.

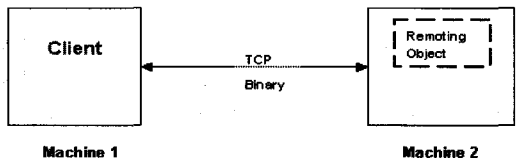
서버에서 클라이언트로 객체를 전달하는 방법은 두 가지로 나뉘는데, 하나는 참조에 의한 전달(Marshal By Reference)이고 다른 하나는 값에 의한 전달(Marshal By Value)이다. 전자는 클라이언트가 원격 객체에 대한 참조 값만으로 그 객체의 함수를 호출하는 방식이며, 클라이언트와 원격 객체가 같은 응용 프로그램 도메인에 있을 경우에만 사용할 수 있다. 후자는 원격 객체 자체를 넘기는 방법으로 클라이언트에 복사본이 존재하게 되며, 클라이언트와 원격 객체가 다른 응용 프로그램 도메인이 있을 경우에는 이 방식을 사용해야 한다.

.NET Remoting에서 사용되는 원격 객체는 크게 서버 활성화 객체(Server Activate Object)와 클라이언트 활성화 객체(Client Activate Object)로 나뉜다. 서버 활성화 객체는 서버에서 활성화되는 객체로서, 단일 호출(Single Call) 객체와 단일 항목(Singleton) 객체가 이에 속한다. 단일 호출 객체는 하나의 클라이언트에만 사용될 수 있고 한 개의 요청에만 응답하는 객

체며, 단일 항목 객체는 여러 클라이언트에 사용될 수 있고 클라이언트 호출 사이에 상태를 공유할 수 있다. 반면에, 클라이언트 활성화 객체는 임대 기반 수명 관리자에 의해 원격 객체의 수명이 관리되며, 클라이언트가 원격 객체를 참조하는 시점에 객체가 생성되고 참조를 마치는 시점에 객체가 소멸된다.

채널은 원격 객체를 대상으로 하는 메시지 전송에 사용된다. 클라이언트가 원격 객체의 함수를 호출하면 호출과 관련된 다른 정보뿐만 아니라 매개 변수도 채널을 통해 원격 객체로 전송되고 호출 결과도 같은 방법으로 클라이언트에게 반환된다. .Net Romoting은 기본적으로 원격 객체와의 통신을 위해 SOAP 프로토콜 또는 HTTP 프로토콜을 사용하는 HTTP 채널과 TCP 프로토콜을 사용하는 TCP 채널을 제공한다.

<그림 4>는 .Net Remoting 클라이언트에서 TCP 채널을 이용해 다른 시스템의 Remoting 객체를 호출하는 모습이다.



<그림 4> 원격 Remoting 객체 호출

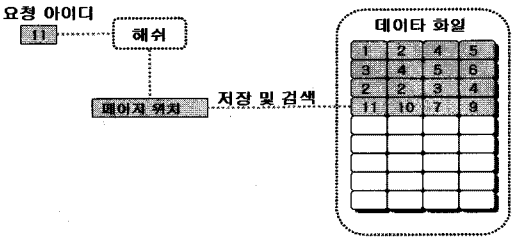
3. 위치 데이타 관리 기법

본 장에서는 대용량 이동 객체 위치 데이타 관리 시스템에서 이동 객체의 위치 데이타를 효율적으로 관리하기 위해서 사용한 해쉬(Hash) 기법, 클러스터링(Clustering) 기법, 시간 질의 검색 기법에 대해서 설명한다.

3.1 해쉬

본 논문에서는 대용량의 이동 객체 위치 데이타를 데이타 화일에 효율적으로 저장 및 검색하기 위해서, 동일한 아이디를 가지는 연속된 일정 개수의 이동 객체의 위치 데이타로 구성된 페이지 단위로 데이타 화일에 저장 및 검색한다. 따라서, 빠른 데이타 화일 접근을 위해서는 대용량의 데이타 화일에서 사용자가 저장 및 검색할 페이지에 빠르게 접근할 수 있어야 한다.

본 논문에서 사용하는 해쉬는 데이타 화일에서 저장 및 검색하는 속도를 향상시키기 위해 사용자가 저장 및 검색하는 이동 객체의 아이디를 키 값으로 하여 사용자가 접근할 페이지 위치 정보를 반환해 준다. 즉, <그림 5>와 같이 이동 객체의 아이디를 이용해 얻어지는 페이지 위치 정보를 가지고 데이타 화일에 직접 접근이 가능하게 된다.



<그림 5> 해쉬 예

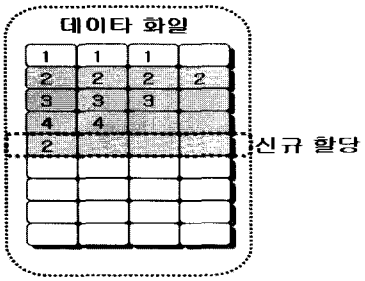
본 논문에서 해쉬는 특정 이동 객체의 아이디를 이용해 직접 접근이 필요한 여러 정보 즉, 데이타 화일의 페이지 위치 정보, 이동 객체의 MBR(Minimal Boundary Rectangle), 이동 객체의 시간 영역(보고된 처음과 마지막 시간)을 관리하는데 사용되고 있다.
또한, 본 논문에서 사용하는 해쉬는 동적 해쉬로써 이동 객체가 증가 및 감소해도 시스템의 성능이 크게 감소되지 않으며, 접근시간을 일정하게 유지해 준다. 하지만 동적 해쉬를 구성하기 위해 약간의 오버헤드가 소요된다.

3.2 클러스터링

클러스터링 기법은 관련된 데이타를 디스크의 인접한 곳에 위치시킴으로써, 데이타 접근 속도를 향상시키는 기법이다. 본 논문에서는 대용량의 위치 데이타에서 이동 객체의 궤적에 대한 순차적인 접근 속도를 증가시키기 위해, 동일한 아이디를 가지는 이동 객체의 위치 데이타를 데이타 화일의 근접된 곳에 연속적으로 저장하는 아이디 기반 클러스터링 기법을 사용하고 있다.

<그림 6>은 클러스터링 기법을 사용하여 데이타 화일에 이동 객체의 위치 데이타를 저장하는 모습을 보여준다. <그림 6>에서 보듯이 모든 이동 객체는 클러스터 크기인 4개의 연속된 페이지를 할당하여 저장되며, 아이디가 2인 이동 객체의 위치 데이타를 저장하

기 위해 데이타 화일에 공간을 할당할 때 지정된 클러스터의 크기(4)만큼 한꺼번에 할당된다.



<그림 6> 클러스터링 기법 예

클러스터링 기법은 순차 접근을 위해 이동 객체를 저장하는 페이지의 크기를 크게 할 경우에 발생하는 임의 접근 속도 저하 문제를 해결하면서, 순차 접근의 속도를 향상시킨다. 즉, 일반적인 임의의 데이타 접근에서는 하나의 페이지 단위로 접근하고, 특정 이동 객체에 대해서만 순차 접근이 필요한 경우 클러스터 단위로 접근함으로써 다양한 접근 방법에 대해 빠른 접근 속도를 보장할 수 있다.

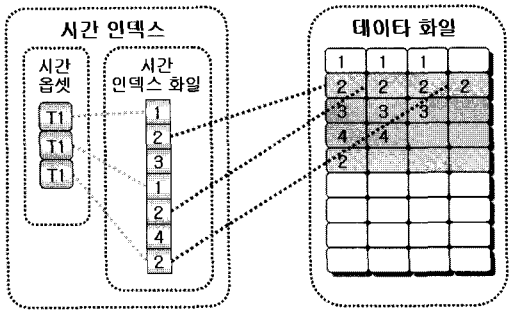
그러나, 클러스터링 기법에서 클러스터 사이즈를 너무 크게 잡았을 경우, 데이타 화일에서 클러스터 크기만큼 미리 할당됨으로써 낭비되는 공간이 증가하게 되므로, 클러스터의 크기 설정 시 주의가 필요하다.

3.3 시간 질의 검색

이동 객체란 객체가 시간에 따라 공간을 이동 하는 객체를 말하며, 이러한 객체의 특성상 일반적인 GIS에서의 공간 질의뿐만 아니라 시간 및 시공간 질의가 요구된다. 따라서, 시간 질의 시 빠른 응답을 위해서는 시간 인덱스가 필요하다.

따라서, 본 논문에서는 해쉬에 기반한 시간 인덱스를 사용해서 데이타 화일에서 사용자가 질의하는 시간에 대한 이동 객체 위치 데이타를 가진 페이지를 빠르게 접근할 수 있다.

시간 인덱스는 해당 시간에 갱신이 발생한 경우, 즉 이동 객체의 위치가 보고된 경우에 해당 이동 객체의 위치 데이타가 저장된 페이지의 위치를 시간 인덱스에 저장한다. 그리고, 사용자가 특정 시간에 속하는 이동 객체의 위치 데이타를 검색할 때, 시간 인덱스에 요청하여 해당 시간의 위치 데이타를 저장한 페이지 위치 정보를 반환받고 이를 이용해 전체 데이타 화일을 검색하지 않고서도 사용자가 질의하는 시간에 이동한 이동 객체에 대한 빠른 응답을 얻을 수 있다. 시간 인덱스의 구성 모습은 <그림 7>과 같다.



<그림 7> 시간 인덱스 구성

<그림 7>에서 보듯이 시간 인덱스는 시간 옵셋과 시간 인덱스 화일로 구성되는데, 시간 옵셋은 해당 시간이 시작되는 시간 인덱스 화일의 옵셋이며, 시간 인덱스 화일은 해당 시간에 갱신된 이동 객체의 아이디와 페이지 위치 정보를 저장한다.

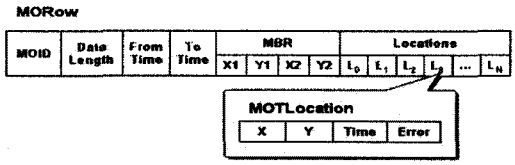
4. 시스템 설계

본 장에서는 대용량 이동 객체 위치 데이타 관리 시스템의 위치 데이타 저장 구조, 디렉토리 및 화일 구조, 전체 시스템 구조, 세부 관리 모듈에 대해서 상세하게 설명한다.

4.1 위치 데이타 저장 구조

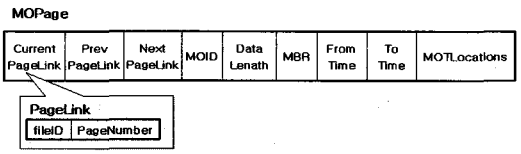
본 시스템에서 저장되는 이동 객체 위치 데이타는 한 시점에 획득된 위치 데이타를 MOTLocation 구조로 나타내는데, MOTLocation에는 2차원 공간상의 x, y 좌표 값, 위치 획득 시간, 위치 획득 오차율이 저장된다.

클라이언트는 관리의 효율성을 위해 MORow 단위로 대용량 이동 객체 위치 데이타 관리 시스템에게 위치 데이타 저장 요청을 하게 되는데, MORow에는 N개의 MOTLocation이 저장된다. <그림 8>은 MORow의 구조를 보여준다.



<그림 8> MORow 구조

대용량 이동 객체 위치 데이타 관리 시스템은 메모리에 이동 객체당 하나의 페이지(MOPage)를 할당하고, 위치 데이타(MOTLocation)를 해당 페이지에 저장한다. MOPage는 데이타 화일에서 해당 페이지가 저장될 위치를 페이지 링크(PageLink) 구조로 나타내는데, PageLink에는 데이타 화일의 아이디와 데이타 화일에서 페이지의 위치를 나타내는 페이지 번호가 저장된다. <그림 9>는 MOPage의 구조를 보여준다.



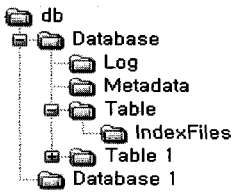
<그림 9> MOPage 구조

4.2 디렉토리 및 화일 구조

본 절에서는 대용량 이동 객체 위치 데이타 관리 시스템에서 이동 객체의 위치 데이타를 저장하는데 사용하는 디렉토리와 화일의 구조에 대해서 설명한다.

4.2.1 디렉토리 구조

대용량 이동 객체 위치 데이타 관리 시스템의 디렉토리 구조는 <그림 10>과 같다.



<그림 10> 디렉토리 구조

<그림 10>에서 보듯이 데이타베이스는 해당 데이타베이스의 이름("Database", "Databaes1")을 가지는 하나의 디렉토리로 구성되며, 관련 화일은 해당 데이타베이스 디렉토리의 하부에 위치한다. 데이타베이스 디렉토리 하부에는 테이블과 관련된 화일을 저장하는 테이블 디렉토리, 로그 화일을 저장하는 로그 디렉토리, 메타데이타 화일을 저장하는 메타데이타 디렉토리가 존재한다.

테이블 디렉토리는 해당 테이블(layer 단위)의 이름("Table", "Table1")을 가지는 하나의 디렉토리로 구성되며, 관련 화일은 해당 데이타베이스 디렉토리의 하부에 위치한다. 테이블 디렉토리 하부에는 테이블과 관련된 데이타 화일과 테이블 메타데이타 화일이 저장되며, 시간별 인덱스 화일을 저장하는 인덱스 디렉토리가 존재한다.

메타데이타 디렉토리는 "Metadata"라는 이름을 가지는 디렉토리로써 하부에 DB 메타데이타 화일이 저장되며, 로그 디렉토리는 "Log"라는 이름을 가지는 디렉토리로써 하부에 로그 화일이 저장된다.

4.2.2 화일 구조

대용량 이동 객체 위치 데이타 관리 시스템에서 사용되는 화일은 데이타를 저장하기 위한 데이타 화일, 데이타베이스와 테이블의 메타데이타 정보를 저장하기 위한 메타데이타 화일, 인덱스를 저장하기 위한 인덱스 화일, 로그를 저장하기 위한 로그 화일이 있다.

가. 데이타 화일

데이타 화일은 이동 객체의 위치 데이타를 저장하는 화일로서, 헤드(Head) 블록과 바디(Body) 블록으로 구성되며, 헤드 블록에는 데이타 화일에 대한 정보가 저장되고 바디 블록에는 이동 객체의 위치 데이타가 페이지 단위로 저장된다. 데이타 화일의 구조는 <그림 11>과 같다.

Head	HeadSize	version	FileID	lastDeletePage
	pageLength	maxDataLength	pageSize	lastPageNumber
Body	page 1	page 2	...	page n

<그림 11> 데이타 화일 구조

나. 인덱스 화일

인덱스 화일은 각 시간 별로 위치가 보고된 이동 객체의 위치 정보를 저장하는 화일로서, 헤드 블록과 바디 블록으로 구성된다. 헤드 블록에는 인덱스 화일에 대한 정보가 저장되고, 바디 블록에는 시간 주기 내에 위치 데이타가 보고된 이동 객체에 대한 페이지 링크

를 가지며, 각 시간 주기에 대한 정보는 인덱스 메타 화일에 저장된다. 인덱스 화일의 구조는 <그림 12>와 같다.

Head	dataLength	dataSize
Body	MOID n	PageLink
	...	...
	MOID m	PageLink
	...	...
	MOID x	PageLink
	...	...
	MOID y	PageLink

<그림 12> 인덱스 화일 구조

또한, 인덱스 메타 화일의 구조는 <그림 13>과 같다.

Head	dataLength	
Body	DateTime	INDEX_OFFSET
	...	...
	DateTime	INDEX_OFFSET

<그림 13> 인덱스 메타 파일

다. 메타데이타 화일

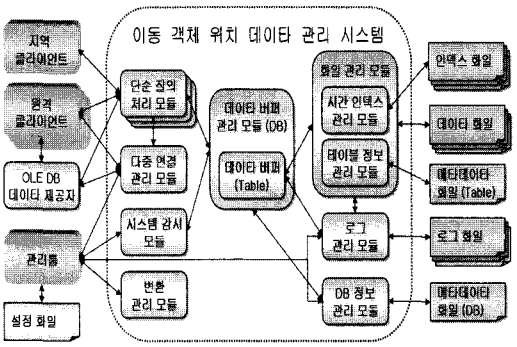
메타데이타 화일은 데이타베이스와 테이블의 정보를 저장하며, XML 화일 형식으로 저장된다. 메타데이타 화일의 내용은 <그림 14>와 같다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<DBMetadata>
  <DBName>testdb</DBName>
  <DBPort>12000</DBPort>
  <DBProcessorLength>10</DBProcessorLength>
  <DBProcessorPorts>12345,12346,12347,12348,12349,12350,12351,12352,12353,12354</DBProcessorPorts>
  <TableLists />
  <MaxDataFileSize>1073741824</MaxDataFileSize>
  <MaxDataLength>2000</MaxDataLength>
  <MaxLogFileSize>5242880</MaxLogFileSize>
  <LogFileName />
  <LogBufferLength>2</LogBufferLength>
- <LogOptions>
  - <LogInsertOptions>
      <Condition>None</Condition>
      <Level>Simple</Level>
    </LogInsertOptions>
  - <LogSearchOptions>
      <Condition>Fail</Condition>
      <Level>Detail</Level>
    </LogSearchOptions>
  </LogOptions>
  <StorageMBR />
  <Shutdown>1</Shutdown>
</DBMetadata>
```

<그림 14> 메타데이타 화일

4.3 전체 시스템 구조

본 논문에서 설계한 대용량 이동 객체 위치 데이타 관리 시스템은 디스크를 이용해 대량으로 발생되는 이동 객체의 위치 데이타를 저장 및 검색하는 시스템이며, 전체 시스템 구조는 <그림 15>과 같다.



<그림 15> 전체 시스템 구조

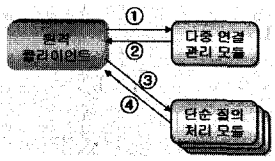
<그림 15>에서 보듯이 대용량 이동 객체 위치 데이타 관리 시스템은 클라이언트에게 접속 가능한 단순 질의 처리 모듈의 접속 정보를 제공해주기 위한다중 연결 관리 모듈, 클라이언트의 질의를 직접 수행하는 단순 질의 처리 모듈, 각 테이블 별로 데이타 버퍼를 생성하고 관리하는 데이타 버퍼 관리 모듈, 데이타 화일, 인덱스 화일, 그리고 각 테이블의 메타데이타 화일을 관리하는 화일 관리 모듈, 위치 데이타 저장 및 검색에 대한 로그를 관리하는 로그 관리 모듈, 데이타베이스의 메타데이타를 관리하는 DB 정보 관리 모듈, 대용량 이동 객체 위치 데이타 관리 시스템의 안정화를 위해 시스템 자원을 감시하는시스템 감시 모듈, 다른 디스크 기반 데이타베이스 시스템과 대용량 이동 객체 위치 데이타 관리 시스템 사이의 데이타 교환을 수행하기위한 변환 관리 모듈, 클라이언트에게 OLE DB 인터페이스를 제공하는 OLE DB 데이타 제공자, 대용량 이동 객체 위치 데이타 관리 시스템을 관리하기 위한 관리툴로 구성된다.

4.4 세부 관리 모듈 설계

본 절에서는 대용량 이동 객체 위치 데이타 관리 시스템을 구성하는 세부 관리 모듈의 설계에 대해서 설명한다.

4.4.1 다중 연결 관리 모듈

다중 연결 관리 모듈은 다중 사용자 연결을 지원하기위한 기능을 수행하며, 단순 질의 처리 모듈과 데이타 버퍼 관리 모듈의 인스턴스(Instance)를 생성한다. 다중 연결 관리 모듈의 동작 과정은 <그림 16>과 같다.



<그림 16> 다중 연결 관리 모듈 내부 동작 모습

<그림 16>에서 보듯이 먼저, 클라이언트가 원격에서 대용량 이동 객체 위치 데이타 관리 시스템의 다중 연결 관리 모듈에 접속하게 되면, 다중 연결 관리 모듈은 현재 동작 가능한 단순 질의 처리 모듈의 접속 정보를 클라이언트에게 반환한다. 클라이언트는 이 정보를 가지고 단순 질의 처리 모듈에 접속하여 저장 및 검색 질의를 전송하면 단순 질의 처리 모듈에서 처리하여 클라이언트에게 결과를 반환하게 된다.

4.4.2 단순 질의 처리 모듈

클라이언트는 다중 연결 관리 모듈을 통해 얻은 채널 정보를 이용해 단순 질의 처리 모듈에 접속한 후, 질의를 요청할 대상 테이블을 선택한다. 단순 질의 처리 모듈은 클라이언트가 요청한 질의를 수행하기 위해 데이타 버퍼 관리 모듈의 함수를 호출하며, 데이타 버퍼 관리 모듈이 질의 처리 결과를 넘겨주면 그 결과를 클라이언트에게 전달한다. 단순 질의 처리 모듈은 이동 객체의 위치 정보 저장 및 검색을 위해 <표 1>과 같은 인터페이스를 제공한다.

<표 1> 이동 객체 관리 인터페이스

질의	내용
Bool InsertRow(MORow moRow)	이동 객체 위치 데이타 삽입
Void Delete (MOID id) Void Delete (MOID id, MOPeriod period) Void Delete (MOPeriod period)	이동 객체 위치 데이타 삭제
IEnumerator IDSearch(MOID[] ids)	주어진 아이디의 궤적 검색
IEnumerator IDSearch(MOID[] ids, MOTime time, MOCapability capability)	주어진 아이디의 지정 시간 이전 또는 이후 위치 데이타 검색
IEnumerator IDSearch(MOID[] ids, MOPeriod period)	주어진 아이디의 지정 시간에 속한 궤적 검색
IEnumerator LastSearch(MOID[] ids)	주어진 아이디의 마지막 위치 데이타 검색
IEnumerator RangeSearch (MOSMBR smbr, MOCapability capability)	주어진 아이디의 지정 지산 및 지정 영역의 궤적 검색

4.4.3 데이타 버퍼 관리 모듈

데이타 버퍼 관리 모듈은 데이타베이스에 존재하는 각 테이블마다 하나의 데이타 버퍼를 생성, 관리, 재구성하는 기능을 수행한다. 또한, DB 정보 관리 모듈과 로그 관리 모듈의 인스턴스를 생성하고 관리한다.

데이타 버퍼 관리 모듈에서는 초기화 함수를 이용하여 기본으로 사용할 테이블을 설정하고, 이 테이블에서 이동 객체의 위치정보를 저장 및 검색할 데이타 버퍼를 생성한다. 또한, 저장 및 검색 요청이 오면, 기본 테이블에 할당된 데이타 버퍼를 활용해 처리한 후 결과를 취합한다. 그리고, 로그 관리 모듈을 통해 처리 결과를 로그에 기록하고, 단순 질의 처리 모듈에 최종 결과를 전달한다.

데이타 버퍼는 메인 메모리에 이동 객체당 하나의 페이지를 할당하여 관리되기 때문에 위치 데이타 저장 시 디스크 접근 횟수를 줄여주며, 위치 데이타 검색 시 최근 데이타에 대한 빠른 검색을 제공한다. 데이타 버퍼에 위치 저장 요청이 들어오면, 먼저 메인 메모리에 존재하는 해당 이동 객체의 페이지에 데이타를 저장한다. 그리고, 페이지에 데이타가 다 차게 되면 화일 관리 모듈에게 해당 페이지의 저장을 요청한다.

데이타 버퍼에 위치 검색 요청이 들어오면, 먼저 메인 메모리에서 데이타를 검색하고, 데이타 화일 검색이 필요한 경우 화일 관리 모듈에게 검색을 요청한다. 그리고 화일 관리 모듈이 검색 결과를 넘겨주면 데이타 버퍼에서 검색된 결과와 취합한 후, 단순 질의 처리 모듈에게 그 결과를 반환한다.

4.4.4 화일 관리 모듈

화일 관리 모듈은 테이블을 생성하거나삭제하고 데이타 화일을 생성하며, 또한 데이타 화일에 저장된 위치 데이타에 대한 다양한 검색 질의를 수행한다. 화일 관리 모듈에서는 위치 데이타의 저장 속도를 높이기 위해 디스크 공간을 미리 할당한 후에 페이지 단위로 저장하는 기능을 제공한다. 또한, MOID 기반 위치 데이타의 검색 속도를 향상시키기 위해 MOID 기반 클러스터 방식을 이용해 페이지가 저장될 공간을 할당하고 저장한다.

<그림 17>은MOID 기반 클러스터 방식으로 페이지가 할당되고 저장되는 모습을 보여준다. <그림 17>에서는 현재 클러스터 크기는 3으로서, 각 MOID 별로 3개의 연속된 페이지를 미리 할당해서 사용하고 있다.

MOID : 3	MOID : 3	MOID : 3	MOID : 5	MOID : 5	MOID : 5
MOID : 4	MOID :4		MOID : 9	MOID : 9	MOID : 9
MOID : 7			MOID : 3		
MOID : 22	MOID : 22		MOID : 35	MOID : 35	MOID : 35

<그림 17> MOID 기반 클러스터 방식

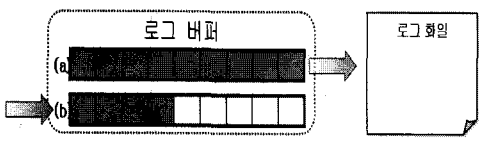
화일 관리 모듈에서는 데이타 화일에서 특정 시간에 해당하는 페이지의 정보를 관리하는 시간 인덱스 관리 모듈과 각 테이블의 정보를 관리하는 테이블 정보 관리 모듈이 포함되어 있다. 시간 인덱스 관리 모듈은 인덱스 화일을 생성하고 이동 객체의 인덱스 정보를 저장, 검색, 삭제하는 기능을 담당한다. 해당 테이블의 데이타 화일에 저장된 모든 이동 객체에 대한 시간 별 인덱스 정보가 인덱스 화일에 저장된다. 테이블 정보 관리 모듈은 해당 테이블의 메타데이타 정보와 데이타 화일 리스트 등을 관리한다.

4.4.5 DB 정보 관리 모듈

DB 정보 관리 모듈은 각 데이타베이스의 메타데이타를 관리하는 기능과 메타데이타 화일에 저장하는 기능을 수행한다. DB 정보 관리 모듈은 XML 화일을 이용해 메타데이타를 저장함으로써 메타데이타의 가독성을 높이며, 사용자의 시스템 환경 설정을 쉽게 한다.

4.4.6 로그 관리 모듈

로그 관리 모듈은 로그 화일을 생성하고, DB 메타데이타에서 설정된 개수만큼의 로그 버퍼를 생성한다. 또한, DB 메타데이타에서 설정된 로그 기록 옵션에 따라 로그를 생성한 후, 이를 로그 버퍼에 저장한다. 사용중인 로그 버퍼가 다 차면, 로그 관리 모듈은 로그 버퍼에 있는 내용을 로그 화일에 기록한다. 로그 관리 모듈에서는 로그 기록을 효율적으로 하기 위해 내부적으로 여러 개의 버퍼를 사용하며, 내부 동작은 <그림 18>과 같다.



<그림 18> 로그 관리 모듈 내부 구조

<그림 18>(a)는 현재 로그 화일에 기록 중이며, <그림 18>(b)는 로그 버퍼에 기록할 때 사용하는 모습을 보여준다. <표 2>는 로그 화일에서 사용하는 로그 옵션을 보여준다.

<표 2> 로그 기록옵션

Select		Insert	
None	Detail	None	Detail
Success	Simple	Success	Simple
Fail		Fail	
All		All	

4.4.7 시스템 감시 모듈

시스템 감시 모듈은 데이타베이스의 시스템 자원을 검사하는 기능을 수행한다. 시스템 감시 모듈은 현재 대용량 이동 객체 위치 데이타 관리 시스템에서 사용하는 CPU, 메모리, 디스크의 용량과 여유 CPU, 메모리, 디스크 용량을 검사하여 반환한다.

4.4.8 OLE DB 데이타 제공자

OLE DB 데이타 제공자는 클라이언트에게 OLE DB 인터페이스를 제공해 주며, 대용량 이동 객체 위치 데이타 관리 시스템에 접속하고 접속 정보를 관리하는 기능을 수행한다. 그리고, 클라이언트로부터 입력된 질의를 분석하여 대용량 이동 객체 위치 데이타 관리 시스템에게 해당 요청을 전달해 주며, 대용량 이동 객체 위치 데이타 관리 시스템으로부터 넘겨받은 처리 결과를 로셋(Rowset) 형태로 구성하여클라이언트에게 반환한다. <표 3>은 로셋 객체로부터 획득할 수 있는 항목들을 보여준다.

<표 3> 로셋 객체가 관리하는 컬럼 정보

컬럼 이름	의미
TableName	질의가 수행된 테이블 명
MOR_MoId	질의 결과 MORow의 Id 값
MOR_DataLength	질의 결과 MORow의 DataLength 값
MOR_FromTIme	질의 결과 MORow의 From 값
MOR_ToTime	질의 결과 MORow의 To 값
MOR_MBRX1	질의 결과 MORow의 Extent 값의 left 값
MOR_MBRY1	질의 결과 MORow의 Extent 값의 bottom 값
MOR_MBRX2	질의 결과 MORow의 Extent 값의 right 값
MOR_MBRY2	질의 결과 MORow의 Extent 값의 top 값
MOR_MOTLocations	질의 결과 데이타의 locations 값

<표 4>는 OLE DB 데이타 제공자가 인식하는 이동 객체의 위치 데이타 저장 및 검색 질의 구문 형식을 보여준다.

<표 4> OLE DB 데이타 제공자의 질의 구문 형식

질의	질의 구문 형식
InsertRow	INSERT INTO *table_name* VALUES (*'id'*, *'locations'*) - 참고: locations는 MOTLocations 값을 순서대로 나열한 것
FirstSearch	SELECT * FROM *table_name* WHERE moID IN (*'id'*, *'id'*,...) AND moTLocations = first(1)
LastSearch	SELECT * FROM *table_name* WHERE moID IN (*'id'*, *'id'*,...) AND moTLocations = last(1)
IDSearch	SELECT * FROM *table_name* WHERE moID = *'id'*
IDSearch	SELECT * FROM *table_name* WHERE moID IN (*'id'*, *'id'*,...)
IDSearch	SELECT * FROM *table_name* WHERE moID IN (*'id'*, *'id'*,...) AND moTLocations.locTime = *'time'*
IDSearch	SELECT * FROM *table_name* WHERE moID IN (*'id'*, *'id'*,...) AND moTLocations.locTime BETWEEN (*'fromtime'*, *'totime'*)
RangeSearch	SELECT * FROM table_name WHERE OverlapFilter *'mbr'* AND BETWEEN (*'fromtime'*, *'totime'*) SELECT * FROM table_name WHERE OverlapRefinement *'mbr'* AND BETWEEN (*'fromtime'*, *'totime'*)
	SELECT * FROM table_name WHERE ContainFilter *'mbr'* AND BETWEEN (*'fromtime'*, *'totime'*) SELECT * FROM table_name WHERE ContainRefinement *'mbr'* AND BETWEEN (*'fromtime'*, *'totime'*)

<표 4>에서 설계된 질의 구문에 나오는 INSERT, SELECT, WHERE, FROM, AND 등의 키워드는 대소문자를 구별하지 않으며, WHERE 절에서 AND를 중심으로 검색 조건의 순서가 바뀌어도 질의 결과는 같다. 그리고, 저장 및 검색 질의에 대한 응답은 성공 또는 실패 코드이며, 검색 질의가 성공적으로 수행된 경우 질의 결과인 MORow는 로셋 객체에 저장된다.

4.4.9 관리툴

관리툴은 대용량 이동 객체 위치 데이타 관리 시스템을 구동시키며, 데이타베이스의 생성, 삭제, 시작, 종료, 그리고 테이블의 생성 및 삭제 기능을 수행한다. 또한, 데이타베이스 목록, 각 데이타베이스의 테이블 목록, 각 테이블의 데이타 화일 목록, 그리고 로그 화일 목록을 보여준다.

관리툴이 시스템을 구동시키면 해당 데이타베이스에 대한 다중 연결 관리 모듈이 생성됨으로써 클라이언트가 원격에서 다중 연결 관리 모듈에 접속할 수 있게 된다.

5. 시스템 구현 및 성능 평가

본 장에서는 대용량 이동 객체 위치 데이타 관리 시스템의 구현에 대해서 살펴보고, 성능 평가에 대해 설명한다.

5.1 시스템 구현 환경

본 논문에서 대용량 이동 객체 위치 데이타 관리 시스템을 구현하기 위해서 운영체제로 Microsoft Windows XP를 사용하였으며, 개발 환경은 Microsoft .NET Framework v1.1.4322을 사용하였다. 그리고, 개발 도구는 Microsoft Visual Studio .NET을 사용하였고, 개발 언어는 C#을 사용하였다.

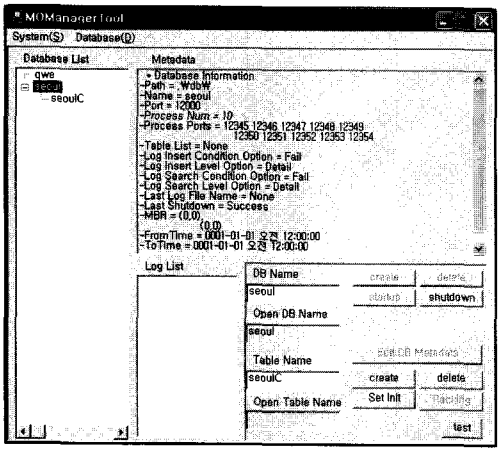
5.2 시스템 구동

대용량 이동 객체 위치 데이타 관리 시스템의 구동 프로그램은 시스템을 구동시키기 위한 관리툴과 시스템에 접속하여 이동 객체의 위치 데이타를 저장 및 검색하기 위한 클라이언트가 있다.

5.2.1 관리툴

관리툴의 사용자 인터페이스는 <그림 21>과 같다. <그림 19>에서 보듯이 Database List는 데이타베이스 목록과 각 데이타베이스 내의 테이블 목록을 트리 형태로 보여주며, Log List는 Database List에서 선택된 데이타베이스의 로그 화일 목록을 보여준다. Metadata는 Database List에서 선택된 데이타베이스의 DB 메타데이타 정보를 보여준다.

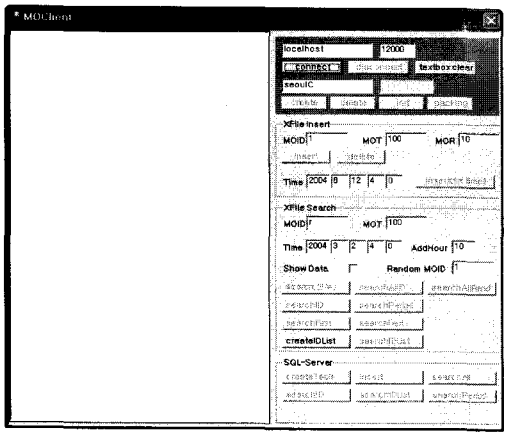
또한, 관리툴은 데이타베이스의 생성, 삭제, 시작, 종료 기능과 테이블의 생성 및 삭제 기능을 제공한다. 데이타베이스의 생성 및 삭제는 시작된 데이타베이스가 없을 경우에만 가능하며, 테이블의 생성 및 삭제는 하나의 데이타베이스가 시작된 경우에만 가능하다.



<그림 19> 관리툴의 사용자 인터페이스

5.2.2 클라이언트

클라이언트는 대용량 이동 객체 위치 데이타 관리 시스템의 성능 평가 및 테스트를 위하여 구현되었다. 클라이언트의 사용자 인터페이스는 <그림 20>과 같다.



<그림 20> 클라이언트의 사용자 인터페이스

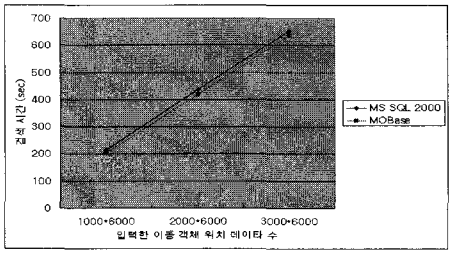
클라이언트 크게 대용량 이동 객체 위치 데이타 관리 시스템에 접속하거나 초기화하기 위한 부분과 테스트할 데이타를 입력하기 위한 부분, 검색 기능을 테스트하기 위한 부분, MS-SQL과의 성능 평가를 위한 부분으로 구성된다.

5.3 성능 평가

본 절에서는 본 논문에서 구현한 대용량 이동 객체 위치 데이타 관리 시스템에 대한 실험 결과를 살펴본다. 즉, 대용량 이동 객체 위치 데이타 관리 시스템(MOBase)의 성능 평가를 위해서 상용 데이타베이스 시스템인 SQL-Server와 비교 실험하였다.

5.3.1 이동 객체 저장 성능 평가

<그림 21>은 이동 객체의 수에 따른 저장 질의 성능을 비교한 그래프이다.



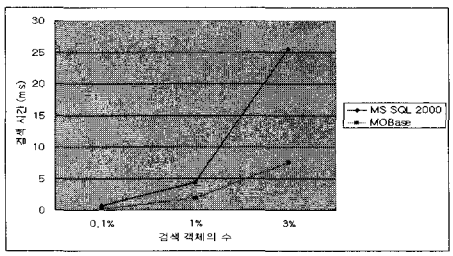
<그림 21> 저장 질의 성능 비교

입력된 이동 객체의 수는 1000, 2000, 3000 개이며, 각 이동 객체는 6000 개의 위치 데이타를 갖는다. 실험 결과를 통해 대용량 이동 객체 위치 데이타 관리 시스템이 SQL-Server 보다 평균 5%의 성능 향상을 보임을 알 수 있다.

5.3.2 이동 객체 검색성능 평가

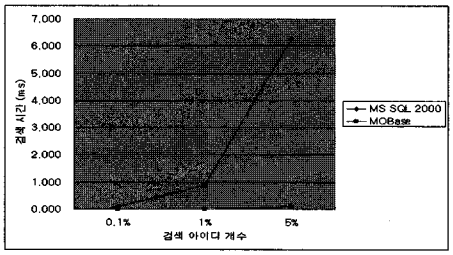
이동 객체 검색 성능 평가에 사용된 이동 객체의 수는 5000개이며, 각 이동 객체는 1분 간격으로 보고된 16000 개의 위치 데이타를 가지고 있다.

<그림 22>는 전체에서 0.1%, 1%, 3%에 해당하는 이동 객체의 모든 위치 데이타를 가져오는 궤적 질의 성능을 비교한 그래프이며, 실험 결과 대용량 이동 객체 위치 데이타 관리 시스템이 평균 3 배의 성능 향상을 보임을 알 수 있다.



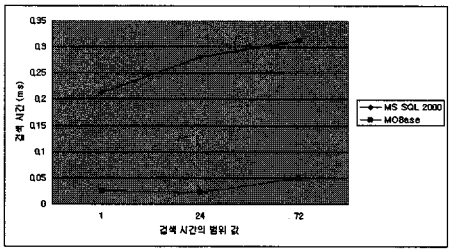
<그림 22> 전체 궤적 질의 성능 비교

<그림 23>은 전체에서 0.1%, 1%, 5%에 해당하는 이동 객체의 가장 마지막 보고된 위치 데이타를 가져오는 질의의 성능을 비교한 그래프이며, 실험 결과 대용량 이동 객체 위치 데이타 관리 시스템이 평균 50배의 성능 향상을 보임을 알 수 있다.



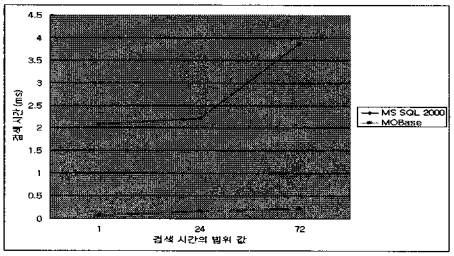
<그림 23> 최근 위치 질의 성능 비교

<그림 24>는 0.1%의 이동 객체에 대해 각각 1, 25, 72 시간 동안의 궤적을 검색하는 질의의 성능을 비교한 그래프이며, 실험 결과를 통해 대용량 이동 객체 위치 데이타 관리 시스템이 평균 8 배의 성능 향상을 보임을 알 수 있다.



<그림 24> 시간 질의 성능 비교

<그림 25>는 1%의 이동 객체에 대해 각각 1, 25, 72 시간 동안의 궤적을 검색하는 질의의 성능을 비교한 그래프이며, 실험 결과를 통해 대용량 이동 객체 위치 데이타 관리 시스템이 평균 18 배의 성능 향상을 보임을 알 수 있다.



<그림 25> 시간 질의 성능 비교

6. 결론

본 논문에서는 이동 객체의 위치 데이타를 보다 효율적으로 관리할 수 있는 대용량 이동 객체 위치 데이타 관리 시스템을 설계 및 구현하였다. 대용량 이동 객체 위치 데이타 관리 시스템은 해쉬, 클러스터링, 시간 질의 검색 기법을 사용하여 대용량의 이동 객체 위치 데이타에 대한 빠른 저장과 검색이 가능하다.

또한, 이동 객체에 대한 궤적 질의와 영역 질의를 제공하며, 불확실한 과거 위치를 추정할 수 있는 기능을 제공한다. 그리고,.Net Remoting을 사용하여 원격에서 대용량 이동 객체 위치 데이타 관리 시스템으로 접속할 수 있으며, OLE DB 인터페이스를 제공한다. 본 논문에서는 대용량 이동 객체 위치 데이타 관리 시스템의 성능 평가를 위해서 상용 데이타베이스 시스템인 SQL-Server와 비교 실험을 수행하였다. 실험 결과를 통해 대용량 이동 객체 위치 데이타 관리 시스템이 SQL-Server보다 이동 객체에 대한 저장 질의에서는 약간의 성능 향상을 보이며, 검색 질의에서는 일반적으로 높은 성능 향상을 보였다.

참고문헌

[1] 박상미, 진희채, 안병익, "위치기반정보서비스를 지원하는 시스템 구조 및 소프트웨어 기술동향 분석," 개방형지리정보시스템학회 학술회의 논문집, Vol.4, No.1, 2001, pp.145-160.

[2] 양영규, "위치기반 서비스(LBS: Location Based Service)기술 현황 및 전망," 정보처리학회지, Vol.8, No.6, 2001, pp.4-5.

[3] 류근호, 안윤애, 이준욱, 이용준, "이동 객체 데이타베이스와 위치 기반 서비스의 적용," 한국정보과학회 데이타베이스 연구, Vol.17, No.3, 2001, pp.57-74.

[4] 장유정, 김동오, 윤재관, 장인성, 한기준, "화일 기반 이동 객체 저장 컴포넌트의 설계 및 구현", 개방형지리정보시스템학회 학술대회 논문집, Vol.5, No.2, 2003, pp.84-90.

[5] 장유정, 김동오, 홍동숙, 한기준, "이동 객체의 효율적인 저장과 검색을 위한 화일 기반 이동 객체 저장 컴포넌트의 개발," 한국정보과학회 학술발표

논문집, Vol.31, No.1, 2004, pp.118-120.

[6] Jensen, C.S., Friis-Christensen, A., Pedersen, T.B., Pfoser, D., Šaltenis, S., and Tryfona, N., "Location-Based Services: A Database Perspective," Proceedings of the 8th Scandinavian Research Conference on Geographical Information Science (ScanGIS), 2001, pp.59-68.

[7] Theodoridis, Y., "Ten Benchmark Database Queries for Location-based Services," The Computer Journal, Vol.46, No.6, 2003, pp.713-725.

[8] Song, Z., and Roussopoulos, N., "Hashing Moving Objects," Proceedings of ACM SIGMOD Int'l Conference on Mobile Data Management, 2001, pp.161-172.

[9] Pfoser, D. and Jensen, C.S., "Capturing the Uncertainty of Moving Object Representations,"Proceedings of the 6th Int'l Symposium on Spatial Databases (SSD), 1999, pp.111-132.

[10] Wolfson, O., Chamberlain, S., Dao S., Jiang, L., and Mendez, G., "Cost and Imprecision in Modeling the Position of Moving Objects," Proceedings of the 14th IEEE Conference on Data Engineering (ICDE), 1998.

[11] Wolfson, O., "Moving Objects Information Management: The Database Challenge," Proceedings of the 5th Workshop on Next Generation Information Technologies and Systems (NGITS), 2002.

[12] Erwig, M., Güting, R. H., Schneider, M., and Vazirgiannis, M., "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases,"GeoInformatica, Vol.3, No.3, 1999, pp.269-296.

[13] Theodoridis, Y., Silva, R., and Nascimento, M., "On the Generation of Spatiotemporal Datasets,"Proceedings of the 6th Int'l Symposium on Spatial Databases (SSD), 1999, pp.147-164.

[14] Forlizzi, L., Güting, R.H., Nardelli, E., and Schneider, M., "A Data Model and Data Structures for Moving Objects Databases," Proceedings of ACM SIGMOD Int'l Conf. on Management of Data, 2000, pp.319-330.

[15] Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., and Vazirgiannis, M., "A Foundation for Representing and Querying Moving Objects," ACM Transactions on Database System, Vol.25, No.1, 2000, pp.1-42.

[16] Sistla, A.P., Wolfson, O., Chamberlain, S., and Dao, S., "Modeling and Querying Moving Objects," Proceedings of the 13th IEEE Conference on Data Engineering (ICDE), 1997, pp.422-432.

[17] 전봉기, 홍봉희, "이동체 색인을 위한 시간 기반 R-트리의 설계 및 구현," 한국정보과학회 논문지, Vol.30, No.3, 2003, pp.320-335.

[18] Pfoser, D., Jensen, C.S., and Theodoridis, Y., "Novel Approaches to the Indexing of Moving Object Trajectories," Proceedings of the 26th Int'l Conference on Very Large Databases(VLDB), 2000, pp.395-406.

[19] Srinivasan, P., An Introduction to Microsoft .NET Remoting Framework, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/introremoting.asp, 2001.

김동오

2000년 건국대학교 컴퓨터공학과 졸업 (공학사)

2002년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)

2002년 ~ 현재 건국대학교 대학원컴퓨터공학과 박사과정

관심분야 : GIS, GML, LBS, MODB, 유비쿼터스

주성완

2004년 건국대학교 컴퓨터공학과 졸업 (공학사)

2004년 ~ 현재 건국대학교 대학원 컴퓨터공학과 석사과정

관심분야 : LBS, MODB, 텔레매틱스

장인성

2001년 부산대학교 전산학과 졸업 (이학석사)

2001년 ETRI 컴퓨터소프트웨어 연구소 GIS 연구팀

2003년 ETRI 컴퓨터소프트웨어 연구소 LBS 연구팀

2004년 ~ 현재 ETRI 텔레매틱스연구단 텔레매틱스 테스트베드기술센터

관심분야 : 텔레매틱스, GIS, LBS, MODB, 유비쿼터스

한기준

1979년 서울대학교 수학교육학과 졸업 (이학사)

1981년 한국과학기술원 전산학과 졸업 (공학석사)

1985년 한국과학기술원 전산학과 졸업(공학박사)

1990년 Stanford 대학 전산학과 visiting scholar

1985년 ~ 현재 건국대학교 컴퓨터공학과 교수

2004년 ~ 현재 한국공간정보시스템학회 회장

2004년 ~ 현재 한국정보시스템감리사협회 회장

관심분야 : 공간데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리