대용량 위치정보 저장시스템 개발†

김동오, 주성완, 홍동숙, 한기준
건국대학교 컴퓨터공학과
{dokim, swju, dshong, kjhan}@db.konkuk.ac.kr

Development of a Storage System for Mass Location Information

Dong-Oh Kim, Sung-Wan Ju, Dong-Sook Hong, Ki-Joon Han
Dept. of Computer Engineering, Konkuk University

요약

최근 이동 객체의 위치정보를 활용한 위치 찾기 서비스, 교통 정보 서비스, 긴급 구조 서비스, 모바일 광고 서비스와 같은 위치 기반 서비스가 부각되고 있다. 이와 같은 다양한 위치 기반 서비스를 제공하기 위해서는 대용량의 이동 객체의 위치정보를 신속하게 저장, 검색, 갱신할 수 있는 저장시스템이 필수적으로 요구된다. 그러나, 이동 객체 위치정보 저장시스템으로 기존의 데이타베이스 시스템을 사용할 경우 불필요한 트랜잭션 연산으로 인하여 저장 및 검색 시 오버헤드가 발생하고, 위치 기반 서비스에 필요한 다양한 질의 및 요구사항을 지원하지 못한다는 문제점이 있다. 따라서, 본 논문에서는 대용량의 위치정보를 효과적으로 저장 및 검색할 수 있으며, 위치 기반 서비스에서 요구하는 궤적 질의 기능, 위치 트리거 기능, 위치 보정 기능, 이동 객체 아이디 기반 클러스터 기능 등을 지원하는 대용량 위치정보 저장시스템을 개발하였다. 또한, 대용량 위치정보 저장시스템의 성능 평가를 위해서 상용 데이타베이스 시스템인 SQL-Server와 비교 실험하여 성능의 우수함을 입증하였다.

1. 서론

최근 이동 객체의 위치정보를 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[10]. 다양한 위치 기반 서비스를 제공하기 위해서는 기하 급수적으로 늘어나는 대용량의 이동 객체의 위치정보를 신속하게 저장하고 검색할 수 있는 이동 객체 데이타베이스 시스템이 필수적으로 요구된다[6, 9, 11].

기존의 상용 데이타베이스 시스템은 이동 객체를 처리하기 위한 시스템이 아니므로 이동 객체 처리 시 불필요한 트랜잭션 연산으로 인해 저장 및 검색 오버헤드가 발생하게 된다. 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 위치정보로 인해 발생되는 이동 객체의 불확실성(Uncertainty) 문제를 처리하지 못하고 있다[8].

이러한 문제점을 해결하기 위해서 본 논문에서는 대용량의 이동 객체 위치정보를 효과적으로 저장하고 검색할 수 있으며, 시공간 질의 기능, 불확실한 과거 위치정보 처리 기능, 위치 트리거 기능, 이동 객체 아이디 기반 클러스터 기능 등을 제공하는 대용량 위치정보 저장시스템을 개발하고 성능 평가를 수행하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 이동 객체와 .NET Remoting에 대해 살펴본다. 제 3장과 제 4장에서는 대용량 위치정보 저장시스템의 설계 및 구현에 대해 상세히 설명하고, 제 5장에서는 실험 및 성능 평가 결과를 분석한다. 마지막으로, 제 6장에서는 결론을 언급한다.

2. 관련 연구

본 장에서는 이동 객체에 대해 살펴보고, 대용량 위치정보 저장시스템에서 클라이언트의 원격 접속을 지원하기 위해 사용된 .Net Remoting에 대해서 알아본다.

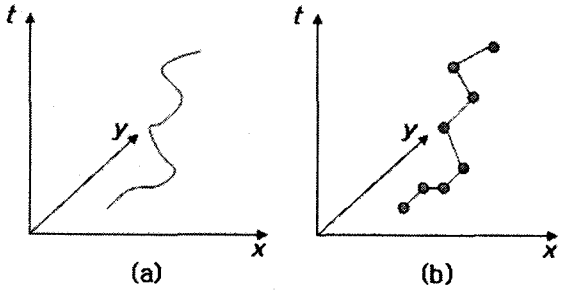
2.1 이동 객체

이동 객체란 시간의 흐름에 따라 연속적으로 위치와 모양이 변하는 시공간 객체를 말하며, 크게 이동 점(moving point)과 이동 영역(moving region)으로 나뉜다[2]. 이동 점은 시간에 따라 위치가 변하는 객체로서 그 예로는 사람, 동물, 자동차, 선박, 비행기 등이 있다. 이동 영역은 시간에 따라 위치와 모양이 변하는 객체로서 그 예로는 태풍, 기온 변화, 암세포 등을 들 수 있다. 본 논문에서 구현한 대용량 위치정보 저장시스템에서는 이동 객체를 이동 점으로 제한한다.

이동 객체를 표현하는 데이타 모델에 관한 기존 연구는 크게 두 가지로 분류된다. 하나는 이동 객체의 과거 위치와 현재 위치를 다루는 데

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성.지원사업의 연구결과로 수행되었음.

이타 모델로서, 이동 객체에 대한 궤적(Trajectory) 질의가 가능하다[1, 2, 3, 4]. 이 데이타 모델은 추상적(Abstract) 모델과 이산적(Discrete) 모델로 나뉘는데, 전자는 그림 1 (a)와 같이 이동 객체의 궤적을 무한한 점들의 집합으로 나타내며, 이를 3차원 공간상의 연속적인 곡선(Curve)으로 표현한다. 반면에, 후자는 그림 1 (b)와 같이 이동 객체의 궤적을 유한한 점들의 집합으로 나타내며, 이를 3차원 공간상의 선분(PolyLine)으로 표현한다. 본 논문에서는 이산적 모델과 같이 이동 객체의 궤적을 유한한 점들의 집합으로 나타낸다.



<그림 1> 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[7].

2.2 .NET Remoting

.NET Remoting은 .NET 프레임워크를 통해 다양한 응용 프로그램 도메인, 프로세스, 컴퓨터에 있는 객체들이 서로 통신할 수 있도록 해주는 원격 기술이다[5]. 또한 .NET 프레임워크는 원격 응용 프로그램에서 메시지를 주고받는 통신 채널뿐만 아니라 활성화 및 수명 지원을 포함한 다양한 서비스를 제공한다. 본 논문에서 구현한 대용량 위치정보 저장시스템은 .NET Remoting을 사용하여 클라이언트의 원격 접속을 지원하고 있다.

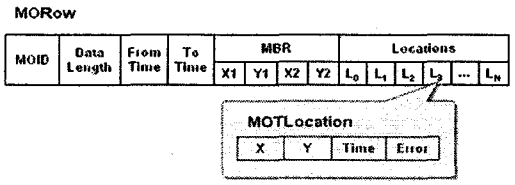
서버에서 클라이언트로 객체를 전달하는 방법은 두 가지로 나뉘는데, 하나는 참조에 의한 전달(Marshal By Reference)이고 다른 하나는 값에 의한 전달(Marshal By Value)이다. 그리고, 채널이 원격 객체를 대상으로 하는 메시지 전송에 사용된다. 클라이언트가 원격 객체의 함수를 호출하면 호출과 관련된 다른 정보뿐만 아니라 매개 변수도 채널을 통해 원격 객체로 전송되고 호출 결과도 같은 방법으로 클라이언트에게 반환된다. .Net Remoting은 기본적으로 원격 객체와의 통신을 위해 SOAP 프로토콜 또는 HTTP 프로토콜을 사용하는 HTTP 채널과 TCP 프로토콜을 사용하는 TCP 채널을 제공한다.

3. 시스템 설계

본 장에서는 대용량 위치정보 저장시스템의 위치정보 저장 구조와 전체 시스템 구조에 대해서 상세하게 설명한다.

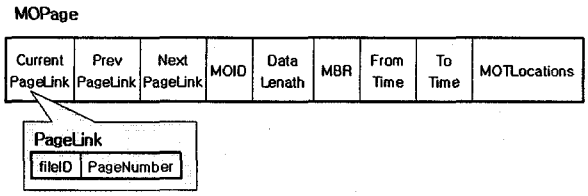
3.1 위치정보 저장 구조

본 시스템에서 저장되는 이동 객체의 위치정보는 한 시점에 획득된 위치정보를 MOTLocation 구조로 나타내는데, MOTLocation에는 2차원 공간상의 x, y 좌표 값, 위치 획득 시간, 위치 획득 오차율이 저장된다. 클라이언트는 관리의 효율성을 위해 MORow 단위로 위치정보를 저장하게 되는데, MORow에는 N개의 MOTLocation이 저장된다. 그림 2는 MORow의 구조를 보여준다.



<그림 2> MORow 구조

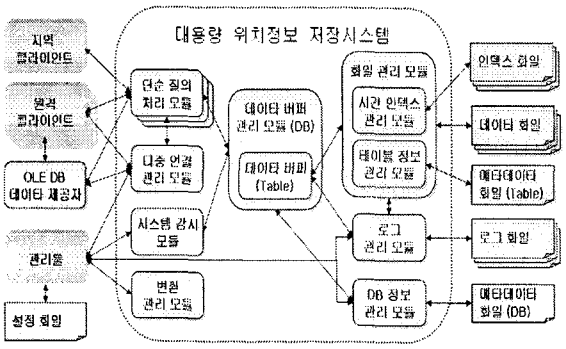
대용량 위치정보 저장시스템은 메모리에 이동 객체당 하나의 페이지(MOPage)를 할당하고, 위치정보(MOTLocation)를 해당 페이지에 저장한다. MOPage는 데이타 화일에서 해당 페이지가 저장될 위치를 페이지 링크(PageLink) 구조로 나타내는데, PageLink에는 데이타 화일의 아이디와 데이타 화일에서 페이지의 위치를 나타내는 페이지 번호가 저장된다. 그림 3은 MOPage의 구조를 보여준다.



<그림 3> MOPage 구조

3.2 전체 시스템 구조

본 논문에서 설계한 대용량 위치정보 저장시스템은 디스크를 이용해 대량으로 발생되는 이동 객체의 위치정보를 저장 및 검색하는 시스템이며, 전체 시스템 구조는 그림 4와 같다.



<그림 4> 전체 시스템 구조

3.2.1 다중 연결 관리 모듈

다중 연결 관리 모듈은 다중 사용자 연결을 지원하기 위한 기능을 수행하며, 단순 질의 처리 모듈과 데이타 버퍼 관리 모듈의 인스턴스(Instance)를 생성한다. 클라이언트가 원격에서 대용량 위치정보 저장시스템의 다중 연결 관리 모듈에 접속하게 되면, 다중 연결 관리 모듈은 현재 동작 가능한 단순 질의 처리 모듈의 접속 정보를 클라이언트에게 반환한다.

3.2.2 단순 질의 처리 모듈

단순 질의 처리 모듈은 클라이언트의 저장 및 검색 질의를 처리하기 위한 모듈로서, 클라이언트가 단순 질의 처리 모듈에 접속한 후, 이동 객체의 위치정보를 저장 및 검색하게 된다. 단순 질의 처리 모듈은 클라이언트가 요청한 질의를 수행하기 위해 데이타 버퍼 관리 모듈의 함수를 호출하며, 데이타 버퍼 관리 모듈에서 처리한 결과를 수집하여 클라이언트에게 전달한다.

3.2.3 데이타 버퍼 관리 모듈

데이타 버퍼 관리 모듈은 저장시스템에 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. 다음, 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.

3.2.4 화일 관리 모듈

화일 관리 모듈은 이동 객체가 저장될 데이타 화일을 생성 및 삭제하며, 데이타 화일에 저장된 위치정보에 대한 다양한 검색 질의를 수행한다.

화일 관리 모듈에서는 위치정보의 저장 속도를 높이기 위해 디스크 공간을 미리 할당한 후에 페이지 단위로 저장하며, 이때 특정 이동 객체의 전체 궤적 질의의 검색 성능을 향상시키기 위해 아이디 기반 클러스터 방식을 사용한다. 아이디 기반 클러스터 방식은 데이타 화일에서 이동 객체의 페이지 할당 시 클러스터 크기만큼 연속으로 할당하는 방식을 말한다.

또한, 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 시간 인덱스 정보를 저장, 검색, 삭제하는 기능을 담당하며, 테이블 정보 관리 모듈은 해당 테이블의 메타데이타 정보와 데이타 화일 리스트 등을 관리한다.

3.2.5 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.

3.2.6 로그 관리 모듈

로그 관리 모듈은 로그 화일을 생성하고, DB 메타데이타에서 설정된 개수만큼의 로그 버퍼를 생성하고 관리한다. 또한, DB 메타데이타에서 설정된 로그 기록 옵션에 따라 로그를 생성한 후, 이를 로그 버퍼에 저장한다. 로그 관리 모듈에서는 로그 기록을 효율적으로 하기 위해 내부적으로 여러 개의 버퍼를 사용한다. 사용중인 로그 버퍼가 다 차면, 로그 관리 모듈은 로그 버퍼에 있는 내용을 로그 화일에 기록한다. 표 1은 로그 관리 모듈에서 사용하는 로그 옵션을 보여준다.

<표 1> 로그 기록 옵션

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

3.2.7 시스템 감시 모듈

시스템 감시 모듈은 데이타베이스의 시스템 자원을 검사하는 기능을 수행한다. 시스템 감시 모듈은 현재 대용량 위치정보 저장시스템에서 사용하는 CPU, 메모리, 디스크의 용량과 여유 CPU, 메모리, 디스크 용량을 검사하여 대용량 위치정보 저장시스템에 반환한다.

3.2.8 OLE DB 데이타 제공자

OLE DB 데이타 제공자는 클라이언트에게 OLE DB 인터페이스를 제공해 주며, 대용량 위치정보 저장시스템에 접속하고, 접속 정보를 관리하는 기능을 수행한다. 그리고, 클라이언트에서 입력된 질의를 분석하여 대용량 위치정보 저장시스템에게 해당 요청을 전달해 주며, 대용량 위치정보 저장시스템으로부터 넘겨받은 처리 결과를 로셋(Rowset) 형태로 구성하여 클라이언트에게 반환한다

3.2.9 관리툴

관리툴은 대용량 위치정보 저장시스템을 구동시키며, 데이타베이스의 생성, 삭제, 시작, 종료 기능과 테이블의 생성 및 삭제 기능, 데이타베이스 목록 검색 기능, 각 데이타베이스의 테이블 목록 및 로그 화일 목록 검색 기능, 시스템 감시 기능 등을 제공한다.

4. 시스템 구현

본 장에서는 대용량 위치정보 저장시스템의 구현 환경을 살펴보고, 시스템의 상세 구현에 대해 설명한다.

4.1 시스템 구현 환경

본 논문에서 대용량 위치정보 저장시스템을 구현하기 위해서 운영체제로 Microsoft Windows XP를 사용하였으며, 개발 환경으로 Microsoft .NET Framework v1.1.4322을 사용하였다. 그리고, 개발 도구는 Microsoft Visual Studio .NET을 사용하였고, 개발 언어는 C#을 사용하였다.

4.2 시스템 상세 구현

본 절에서는 대용량 이동 객체 위치정보 관리 시스템의 세부 모듈 구현에 대해서 상세하게 설명한다.

4.2.1 다중 연결 관리 모듈

다중 연결 관리 모듈의 주요 함수는 표 2와 같다.

<표 2> 다중 연결 관리 모듈의 주요 함수

질의	내용
public string Startup(string path, string name)	다중 연결 관리 모듈 구동
private int LocalCMStart(int port)	다중 연결 관리 모듈 채널 생성
private int LocalSQPStart(int port)	단순 질의 처리 모듈 채널 생성

Startup 함수는 데이타 버퍼 관리 모듈의 인스턴스를 생성한 후, 클라이언트와의 연결을 위해 .Net Remoting 클래스인 MOConnectionAPI와 MOSimpleQueryProcessor의 인스턴스를 생성한다. LocalCMStart 함수는 클라이언트가 접속할 수 있는 채널을 열어주며, LocalSQPStart 함수는 DB 메타데이타에서 설정된 프로세스 개수만큼 단순 질의 처리 모듈을 구동시킨다.

4.2.2 단순 질의 처리 모듈

단순 질의 처리 모듈은 사용자의 질의를 받아 처리하는 모듈로서 이 모듈의 주요 인터페이스는 표 3과 같다.

<표 3> 이동 객체 관리 인터페이스

질의	내용
public bool Initialize(string layerName)	시스템 기능 설정
public void Packing(bool isContinue)	데이타 화일 Packing
public void Backup()	데이타 백업
public MOSMBR StorageMBR()	저장된 모든 이동 객체의 MBR 및 시간 범위 값 반환
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

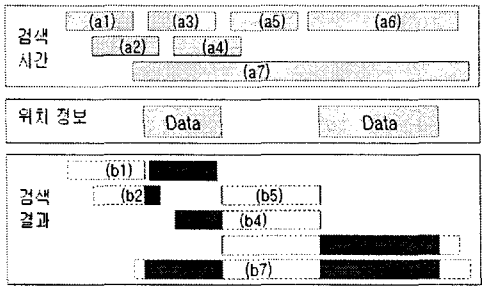
4.2.3 데이타 버퍼 관리 모듈

데이타 버퍼 관리 모듈은 DB 정보 관리 모듈을 생성하고, DB 정보 관리 모듈을 통해 로그 화일의 이름과 로그 기록 옵션 정보를 가져오며, 이 정보를 이용해 로그 관리 모듈을 생성한다. 또한, 데이타 버퍼 관리 모듈은 단순 질의 처리 모듈의 요청에 따라 이동 객체의 위치정보를 저장하고 검색하는 기능을 제공한다. 데이타 버퍼 관리 모듈의 주요 함수는 표 4와 같다.

<표 4> 데이타 버퍼 관리 모듈의 주요 함수

질의	내용
public MODataBufferManager()	데이타 버퍼 관리 모듈 생성자
public string Startup(string dbPath, string dbName, char startType, bool reinsert)	데이타 버퍼 관리 모듈 구동
public string Shutdown()	데이타 버퍼 관리 모듈 종료
public bool Initialize(string layerName)	시스템 기능 설정
public void Packing(bool isContinue)	데이타 화일 Packing
public void Backup()	데이타 백업
public MOSMBR StorageMBR()	저장된 모든 이동 객체의 MBR 및 시간 범위 값 반환
public string CreateTable(string layerName)	테이블 생성
public string DeleteTable(string layerName)	테이블 삭제
public ArrayList GetTableList()	데이타 목록 반환
public bool TestTrigger(MORow moRow)	트리거 테스트
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

표 4에서 궤적 질의는 사용자가 요청한 시간에 보고된 위치정보가 존재하지 않을 경우 불확실성 문제가 발생하게 된다. 본 시스템에서는 이를 해결하기 위해 사용자가 질의하는 시간에 보고된 데이타가 없을 경우 이를 보정할 수 있도록 해당 시간을 벗어난 가장 최초의 값을 이용하게 된다. 그림 5는 사용자의 요청한 시간과 결과 데이타의 모습을 보여준다.



<그림 5> 시간 질의 결과 예

그림 5에서 검색 시간 (a5)일 경우 사용자가 질의한 시간에 저장된 위치정보가 존재하지 않지만, (b5)와 같이 위치 보정을 위하여 해당 시간과 가장 가까운 위치정보를 검색 결과로 반환하여 준다.

그리고, 대용량으로 발생되는 위치정보 저장 시 효율성을 증대시키기 위하여, 본 시스템에서는 메모리의 페이지를 이용해 위치정보를 저장하고, 페이지가 꽉 찼을 경우에만 데이타 화일에 기록한다. 그림 6은 본 시스템에서 사용하는 위치정보 저장 알고리즘을 보여준다.

```
public string Insert(MORow moRow)
{
  moPage = LocalFindPage(moRow.ID); // 해당 MOID가 저장된 페이지 검색
  If(moPage == null){ // 해당 페이지 없을 경우 새로운 페이지 할당
    moPage = new MOPage(moRow.ID, maxDataLength);
    newPageLink = fileManager.GetLastPageLink(); // 데이타 화일의 위치 할당
    moPage.currentPageLink = newPageLink;
    LocalAddKey(moPage.moID, moPage); // 페이지 등록
  }
  for(int i=0; i<moRow.DataLength; i++) // 입력한 이동 객체의 위치 데이타를 페이지에 기록
  {
    if( moPage.dataLength == moPage.maxDataLength)
    { // 페이지에 데이타가 다 차면, 현재 페이지를 데이타 화일에 기록하고 새로운 페이지 할당
        newmoPage = new MOPage(moRow.ID, maxDataLength);
        newPageLink = fileManager.GetLastPageLink();
        newmoPage.currentPageLink = newPageLink;
        newmoPage.prevPageLink = moPage.currentPageLink;
        moPage.nextPageLink = newPageLink;
        int update = fileManager.Write(moPage); // 데이타 화일에 현재 페이지 기록
        moPage = newmoPage;
        LocalAddKey(moPage.moID, moPage);
    }
    moT = moRow[i];
    moPage.Insert(moT);  // 페이지에 위치정보 저장
  }
  return null;  // 위치 저장 성공 시 null 값 반환
}
```

<그림 6> 위치 저장 알고리즘

4.2.4 화일 관리 모듈

화일 관리 모듈은 데이타 화일에 접근하는 기능을 전담하는 모듈로서, 화일 관리 모듈의 주요 함수는 표 5와 같다.

<표 5> 화일 관리 모듈의 주요 함수

질의	내용
public MOFileManager(string dbPath, string dbName, uint length, ulong fileSize)	화일 관리자 생성자
public string Startup(string tableName, out long defFileID)	화일 관리자 시작
public int Shutdown()	화일 관리자 종료
public int CreateTable(string tableName)	테이블 생성
public int DropTable(string tableName)	테이블 삭제
private int CreateDataFile()	데이타 화일 생성
private int OpenDataFile(string fileName)	데이타 화일 오픈
public int Write(MOPage page)	데이타 화일에 페이지 기록
public int WritePageLink(MOPage page)	데이타 화일의 페이지 링크기록
private MOPage ReadPage(PageLink pLink)	해당 페이지 링크의 페이지 읽기
public PageLink GetLastPageLink()	데이타 화일의 새 페이지 링크 획득
public MOTLocation GetFirstMOTLocation(PageLink pageLink)	해당 페이지의 첫번째 위치 데이타 획득
public MOTLocation GetLastMOTLocation(PageLink pageLink)	해당 페이지의 마지막 위치 데이타 획득
public MORow SearchAll(MOID moID, PageLink firstLink, PageLink lastLink, uint length)	두 페이지 링크 사이에 있는 모든 위치 데이타 획득
public MOMBR GetMBR(PageLink firstLink, PageLink lastLink)	두 페이지 링크 사이에 있는 모든 위치 데이타 의 MBR 검색
public MORow PreviousSearch(MOID moID, PageLink firstLink, PageLink lastLink, MOTime fromTime, uint length)	두 페이지 링크 사이에서 지정 시간 이전의 첫번째 위치 데이타 획득
public MORow NextSearch(MOID moID, PageLink firstLink, PageLink lastLink, MOTime fromTime, uint length)	두 페이지 링크 사이에서 지정 시간 이후의 첫번째 위치 데이타 획득

4.2.5 DB 정보 관리 모듈

DB 정보 관리 모듈은 DB 메타데이타 화일과 테이블 메타데이타 화일을 생성 및 저장하고, 모든 메타데이타 정보를 관리하는 기능을 제공한다. DB 정보 관리 모듈의 주요 함수는 표 6과 같다.

<표 6> DB 정보 관리 모듈의 주요 함수

질의	내용
public int WriteDBMetadata()	데이타베이스 메타데이타 기록
private int ReadDBMetadata()	데이타베이스 메타데이타 읽기
public int WriteTableMetadata()	테이블 메타데이타 기록
private int ReadTableMetadata()	테이블 메타데이타 읽기
public int AddTable(string tableName)	테이블 추가
public int RemoveTable(string tableName)	테이블 삭제
public bool FindTable(string tableName)	테이블 검색
public ArrayList GetTableList()	테이블 목록 검색
public SortedList GetFileList(string tableName)	데이타 화일 목록 검색
public int AddDataFile(string tableName, long fileID, string fileName)	데이타 화일 추가
public int UpdateDataFile(string tableName, long fileID, ulong dataLength, MOMBR mbr, MOTime from, MOTime to)	데이타 화일 정보 갱신

4.2.6 로그 관리 모듈

로그 관리 모듈은 로그 기록에 관련된 기능을 전담하는 모듈로서, 로그 관리 모듈의 주요 함수는 표 7과 같다.

<표 7> 로그 관리 모듈의 주요 함수

질의	내용
public int AddLog(string log)	로그 추가
public int AddLog(string query, bool success, MORow moRow, string err)	로그 추가
public int ChangeLogBuffer()	로그 버퍼 교체
private int WriteLogFile(int logBufferNumber)	로그 버퍼의 내용을 로그 화일에 기록

4.2.7 OLE DB 데이타 제공자

OLE DB 데이타 제공자는 ATL COM을 사용하여 구현되었으며, C#으로 구현된 MOMidCom을 이용해 대용량 위치정보 저장시스템의 함수를 호출한다. 표 8은 OLE DB 데이타 제공자가 인식하는 이동 객체의 위치정보 저장 및 검색 질의 구문 형식을 보여준다.

<표 8> 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 8에서 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. 표 9는 로셋 객체로부터 획득할 수 있는 항목들을 보여준다.

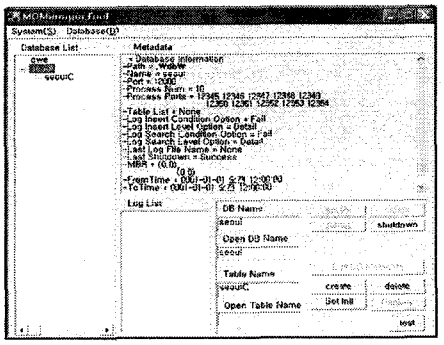
<표 9> 로셋 객체가 관리하는 컬럼 정보

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

4.2.8 관리툴

관리툴의 인터페이스는 그림 7과 같다. 그림에서 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 기능 데이타베이스 목록 검색 기능, 각 데이타베이스의 테이블 목록 및 로그 화일 목록 검색 기능, 시스템 감시 기능 등을 제공한다. 데이타베이스의 생성 및 삭제는 시작된 데이타베이스가 없을 경우에만 가능하며, 테이블의 생성 및 삭제는 하나의 데이타베이스가 시작된 경우에만 가능하다.



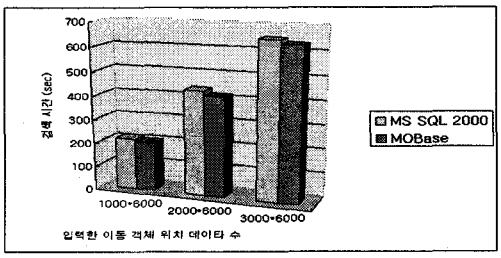
<그림 7> 관리툴의 인터페이스

5. 실험 및 성능 평가

본 장에서는 본 논문에서 구현한 대용량 위치정보 저장시스템에 대한 실험 결과를 살펴본다. 실험을 위해 사용된 시스템의 하드웨어 사양은 Intel Xeon CPU 2.80GHz, 3.87GB RAM이며, 운영체제는 Windows Server 2003을 사용하였다. 그리고, 대용량 위치정보 저장시스템(MOBase)의 성능 평가를 위해서 상용 데이타베이스 시스템인 SQL-Server와 비교 실험하였다.

5.1 이동 객체 저장 성능 평가

그림 8은 이동 객체의 수에 따른 저장 질의 성능을 비교한 그래프이다.



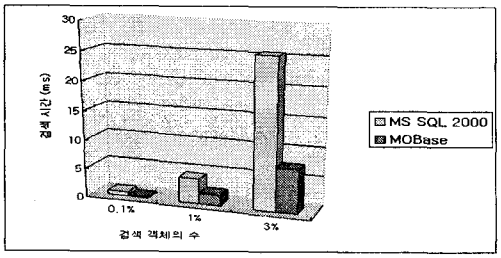
<그림 8> 저장 질의 성능 비교

입력된 이동 객체의 수는 1000, 2000, 3000개이며, 각 이동 객체는 6000 개의 위치정보를 갖는다. 실험 결과를 통해 대용량 위치정보 저장시스템이 SQL-Server 보다 평균 5%의 성능 향상을 보임을 알 수 있다.

5.2 이동 객체 검색 성능 평가

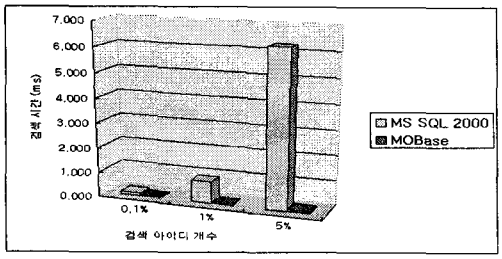
이동 객체 검색 성능 평가에 사용된 이동 객체의 수는 5000개이며, 각 이동 객체는 1분 간격으로 보고된 16000 개의 위치정보를 가지고 있다.

그림 9는 전체에서 0.1%, 1%, 3%에 해당하는 이동 객체의 모든 위치정보를 가져오는 궤적 질의 성능을 비교한 그래프이며, 실험 결과 대용량 위치정보 저장시스템이 평균 3 배의 성능 향상을 보임을 알 수 있다.



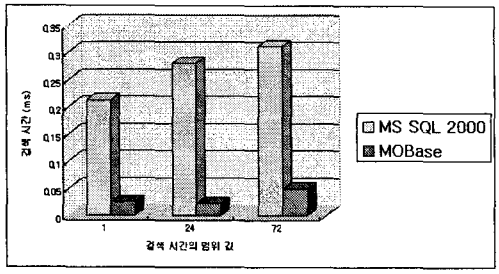
<그림 9> 전체 궤적 질의 성능 비교

그림 10은 전체에서 0.1%, 1%, 5%에 해당하는 이동 객체의 가장 마지막 보고된 위치정보를 가져오는 질의의 성능을 비교한 그래프이며, 실험 결과 대용량 위치정보 저장시스템이 평균 50 배의 성능 향상을 보임을 알 수 있다.



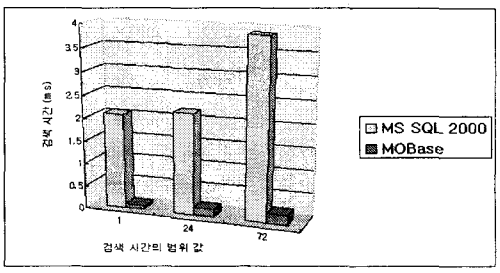
<그림 10> 최근 위치 질의 성능 비교

그림 11은 0.1%의 이동 객체에 대해 각각 1, 25, 72 시간 동안의 궤적을 검색하는 질의의 성능을 비교한 그래프이며, 실험 결과를 통해 대용량 위치정보 저장시스템이 평균 10 배의 성능 향상을 보임을 알 수 있다.



<그림 11> 시간 질의 성능 비교

그림 12는 1%의 이동 객체에 대해 각각 1, 25, 72 시간 동안의 궤적을 검색하는 질의의 성능을 비교한 그래프이며, 실험 결과를 통해 대용량 위치정보 저장시스템이 평균 20 배의 성능 향상을 보임을 알 수 있다.



<그림 12> 시간 질의 성능 비교

6. 결론

본 논문에서는 대용량 이동 객체의 위치정보를 보다 효율적으로 저장 및 검색할 수 있는 대용량 위치정보 저장시스템을 설계 및 구현하였다. 또한, 이동 객체에 대한 궤적 질의와 영역 질의를 제공하며, 불확실한 과거 위치를 추정할 수 있는 기능을 제공한다. 그리고,.Net Remoting을 사용하여 원격에서 대용량 위치정보 저장시스템으로 접속할 수 있으며, OLE DB 인터페이스를 제공한다.

본 논문에서는 대용량 위치정보 저장시스템의 성능 평가를 위해서 상용 데이타베이스 시스템인 SQL-Server와 비교 실험을 수행하였다. 실험 결과를 통해 대용량 위치정보 저장시스템이 SQL-Server보다 이동 객체에 대한 저장 질의에서는 약간의 성능 향상을 보이며, 검색 질의에서는 일반적으로 높은 성능 향상을 보였다.

참고 문헌

[1] Erwig, M., Güting, R. H., Schneider, M., and Vazirgiannis, M., "Abstract and Discrete Modeling of Spatio-Temporal Data Types," Proceedings of the 6th ACM Int'l Workshop on Geographical Information Systems (ACM-GIS), 1998, pp.131-136.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] Wolfson, O., Xu, B., Chamberlain, S., and Jiang, L., "Moving Objects Databases: Issues and Solutions," Proceedings of the 10th Int'l Conference on Scientific and Statistical Database Management (SSDBM), 1998, pp.111-122.

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

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

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

[11] 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.