화일 기반 이동 객체 저장 컴포넌트의 설계 및 구현†

장유정*, 김동오*, 윤재관*, 장인성**, 한기준*
건국대학교 컴퓨터공학과*, 한국전자통신연구원**
*{yjjang, dokim, jkyun, kjhan}@db.konkuk.ac.kr, **e4dol2@etri.re.kr

Design and Implementation of a File-based Moving Object Storage Component

Yu-Jung Jang*, Dong-O Kim*, Jae-Kwan Yun*, In-Sung Jang**, Ki-Joon Han*
Dept. of Computer Engineering, Konkuk University*, ETRI**

요약

최근 이동 컴퓨팅 기술이 급속도로 발전하면서 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 서비스를 제공하기 위해서는 이동 객체의 위치정보를 신속하게 저장, 검색, 갱신할 수 있는 이동 객체 데이타베이스 시스템이 필수적으로 요구된다. 그러나, 기존의 데이타베이스 시스템을 사용할 경우에는 저장 공간을 많이 차지게 되고, 불필요한 트랜잭션 연산으로 인하여 대용량의 위치 데이타 관리에 큰 오버헤드가 발생한다는 문제가 있다. 그러므로, 이동 객체의 위치 데이타를 저장하고 관리하기 위하여 화일 기반의 저장 시스템을 사용하는 것이 보다 더 효율적이다. 이에 본 논문에서는 대용량의 위치 데이타를 효과적으로 저장 및 검색할 수 있는 화일 기반 이동 객체 저장 컴포넌트를 설계하고 구현하였다. 화일 기반 이동 객체 저장 컴포넌트는 다중 연결 관리자, 단순 질의 처리기, 메타데이타 관리자, 데이타 화일 관리자, 인덱스 화일 관리자, 로그 관리자, 관리툴로 구성된다.

1. 서론

최근 이동 컴퓨팅 기술의 발전과 무선 측위 기술의 발달, 그리고 무선 인터넷 사용자의 급증에 따라 이동 단말기의 위치정보를 활용한 위치 기반 서비스(LBS: Location Based Services)가 점차 확대되고 있다. 이에 따라 이동 객체의 위치정보를 효과적으로 처리하고 관리하기 위한 이동 객체 데이타베이스가 요구되고 있다[1,6,7].

이동 객체 데이타베이스에 대한 기존 연구는 크게 두 가지로 구분될 수 있다. 하나는 이동 객체의 현재 위치에 시간 함수를 적용하여 미래 위치를 예측하는 연구이다[4,6]. 다른 하나는 이동 객체의 현재 및 과거 위치를 이용하여 이동 객체가 이동한 궤적을 검색하는 연구이다[1,2]. 그러나, 후자의 경우에는 이동 객체의 궤적을 지속적으로 저장해야 하므로, 이동 객체의 수가 많고 위치 획득 간격이 짧을수록 위치 데이타는 대용량이 되고 이로 인한 저장 및 검색 오버헤드가 증가하게 된다. 본 논문에서는 후자의 경우에 발생하는 문제를 해결하기 위하여 화일 기반 이동 객체 저장 컴포넌트를 설계 및 구현하였다.

이전 연구에서는 대용량의 위치 데이타를 다양한 데이타베이스 시스템에 효과적으로 분산 저장하고 검색할 수 있는 분산 위치 저장 컴포넌트를 개발하였다[7]. 그러나, 기존의 데이타베이스 시스템이 제공하고 있는 트랜잭션 처리와 같은 기능은 이동 객체의 위치정보만을 처리하기 위한 시스템에서는 트랜잭션 연산의 증가로 대용량의 위치 데이타를 처리하기에 비효율적이다. 이에 본 논문에서는 이동 객체를 관리하기 위하여 화일 기반 이동 객체 저장 컴포넌트를 설계 및 구현함으로써 이동 객체에 대한 효과적인 저장 및 검색 기능을 제공한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로 위치정보 관리 시스템과 분산 위치 저장 컴포넌트, 그리고 .NET Remoting에 대해 살펴본다. 제 3 장과 제 4 장에서는 화일 기반 이동 객체 저장 컴포넌트의 설계 및 구현에 대해 상세히 설명하고, 끝으로 제 5 장에서는 결론 및 향후 연구 과제에 대해서 언급한다.

† 본 연구는 한국전자통신연구원의 개방형 LBS 핵심기술개발과제의 연구비 지원에 의해 수행되었음.

2. 관련 연구

본 장에서는 위치정보 관리 시스템과 이전 연구에서 개발한 분산 위치 저장 컴포넌트에 대해서 알아보고, 화일 기반 이동 객체 저장 컴포넌트에서 데이타 전송을 위해 사용된 .Net Remoting에 대해 기술한다.

2.1 위치정보 관리 시스템

위치정보 관리 시스템(LIMS: Location Information Management System)은 한국전자통신연구원(ETRI)에서 다양한 위치 기반 응용 서비스를 제공하기 위해 개발하는 시스템이다[8]. 위치정보 관리 시스템의 전체 구조는 그림 1과 같으며, 크게 위치획득 서브시스템, 위치저장 서브시스템, 위치질의 서브시스템으로 나뉘어진다.

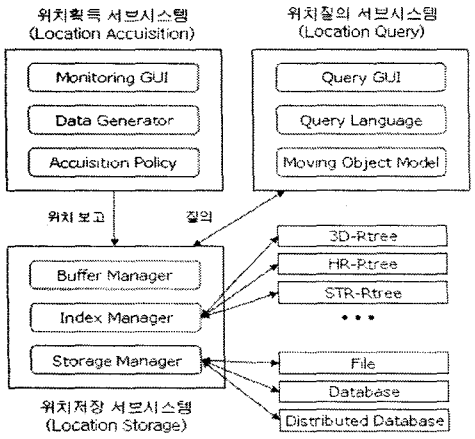

그림 1. 위치정보 관리 시스템

위치획득 서브시스템은 다양한 위치획득 전략에 따라 이동 객체의 현재 위치 데이타를 획득하여 위치저장 서브시스템에게 전해주는 시스템이다. 위치저장 서브시스템은 위치획득 서브시스템으로부터 보고된 위치정보를 메모리 버퍼를 통해 위치 색인과 위치 저장소에 저장하는 시스템으로 버퍼 관리자, 색인 관리자, 저장 관리자로 구성된다. 위치질의 서브시스템은 이동 객체를 표현하는 데이타 구조와 질의 연산자를 정의한 이동 객체 모델을 기반으로 위치질의를 수행한다. 앞에서 설명한 분산 위치 저장 컴포넌트는 위치저장 서브시스템의 저장 관리자에서 분산 위치 저장의 기능을 담당하는 컴포넌트이다.

2.2 분산 위치 저장 컴포넌트

분산 위치 저장 컴포넌트는 .NET 플랫폼 상에서 대용량의 위치 데이타를 여러 데이타베이스 시스템에 효과적으로 분산 저장하고 검색하는 기능을 제공한다[7]. 분산 위치 저장 컴포넌트의 구조는 그림 2와 같다.

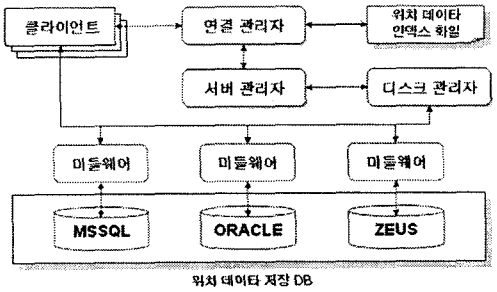

그림 2. 분산 위치 저장 컴포넌트의 구조

분산 위치 저장 컴포넌트에서 미들웨어는 클라이언트의 위치 데이타 저장 및 검색 요청을 수행하며, 서버 관리자는 분산된 미들웨어의 정보(IP, PORT 번호, 서비스 이름)를 유지 및 관리하고, 저장 요청이 발생하면 등록된 미들웨어에 대한 로드 밸런싱을 수행한다. 또한, 연결 관리자는 빠른 검색을 위해 위치 데이타에 대한 인덱스 정보를 메모리 상에서 관리하며, 디스크 관리자는 등록된 서버의 디스크 상태를 주기적으로 확인하고 디스크 사용을 관리한다.

위치 데이타를 검색하기 위해서는 우선 클라이언트가 연결 관리자에게 검색하고자 하는 위치 데이타의 인덱스 정보를 요청한다. 연결 관리자는 해당 위치 데이타가 저장된 미들웨어에 대한 인덱스 정보를 넘겨주면, 클라이언트는 이 정보를 이용해서 미들웨어에 접속하고 위치 데이타 검색을 요청한다. 검색 요청을 받은 미들웨어는 위치 데이타 저장 DB에 질의를 전달하고, 검색이 성공적으로 수행되면 검색 결과를 클라이언트로 반환한다. 만약, 위치 데이타 검색에 실패했을 경우에는 그에 대한 로그 정보를 로그 화일에 기록한다.

위치 데이타를 저장하기 위해서 클라이언트는 연결 관리자를 통해 가장 부하가 적은 미들웨어에 대한 정보를 넘겨받는다. 그리고, 이 정보를 이용해서 해당 미들웨어에 접속한 후 위치 데이타 저장을 요청한다. 저장 요청을 받은 미들웨어는 위치 데이타 저장 DB에 질의를 전달하고, 저장이 성공적으로 수행되면 연결 관리자에게 해당 위치 데이타가 저장된 미들웨어에 대한 정보를 전달하게 된다. 만약, 위치 데이타 저장에 실패한다면 그에 대한 로그 정보를 로그 화일에 기록한다.

2.3 .NET Remoting

.NET Remoting은 .NET 프레임워크를 통해 다양한 응용 프로그램 도메인, 프로세스, 컴퓨터에 있는 개체들이 서로 통신할 수 있도록 해준다[3]. 또한 .NET 프레임워크는 원격 응용 프로그램에서 메시지를 주고받는 통신 채널뿐만 아니라 활성화 및 수명 지원을 포함한 다양한 서비스를 제공한다.

.NET Remoting에서 사용되는 원격 개체의 종류는 크게 서버 활성화 개체(Server Activate Object)와 클라이언트 활성화 개체(Client Activate Object)로 나뉠 수 있다[5]. 서버 활성화 개체는 서버에서 활성화되는 개체로서 단일 호출(Single Call) 개체와 단일 항목(Singleton) 개체가 이에 속한다. 단일 호출 개체는 하나의 클라이언트에만 사용할 수 있고 한 개의 요청에만 응답하는 개체이며, 단일 항목 개체는 여러 클라이언트에 사용할 수 있고 클라이언트 호출 사이에 상태를 공유할 수 있는 개체이다. 반면에, 클라이언트 활성화 개체는 임대 기반 수명 관리자에 의해 원격 개체의 수명이 관리되며, 클라이언트가 원격 개체를 참조하는 시점에 생성되고 참조를 마치는 시점에 소멸되는 개체이다.

3. 시스템의 설계

본 장에서는 화일 기반 이동 객체 저장 컴포넌트의 설계에 대해서 상세하게 설명한다.

3.1 시스템 구조

본 논문에서 설계한 화일 기반 이동 객체 저장 컴포넌트의 구조는 그림 3과 같다. 화일 기반 이동 객체 저장 컴포넌트는 다중 사용자 관리를 위한 다중 연결 관리자, 저장 및 검색 질의를 수행하는 단순 질의 처리기, 최근 데이타를 메모리 상에서 유지하기 위해 해쉬 인덱스를 관리하는 인덱스 관리자, 메타데이타 정보를 관리하는 메타데이타 관리자, 데이타 화일에 이동 객체의 위치 데이타를 저장하고 검색하는 데이타 화일 관리자, 이동 객체에 대한 인덱스 정보를 관리하는 인덱스 화일 관리자, 저장과 검색 질의 수행 결과를 로그로 남기기 위한 로그 관리자, 화일 기반 이동 객체 저장 컴포넌트를 구동시키고 관리하기 위한 관리툴, 그리고 테스트를 위한 클라이언트로 구성된다.

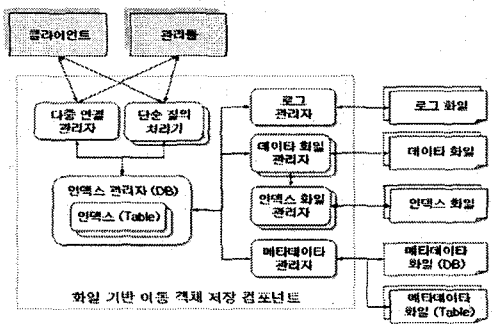

그림 3. 화일 기반 이동 객체 저장 컴포넌트의 구조

3.1.1 이동 객체의 저장 구조

클라이언트는 MORow 단위로 화일 기반 이동 객체 저장 컴포넌트에게 위치 데이타 저장을 요청하는데, MORow의 구조는 그림 4와 같다.

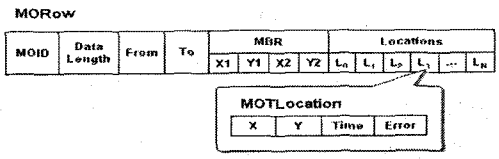

그림 4. MORow 구조

이동 객체 저장 요청이 들어오면 화일 기반 이동 객체 저장 컴포넌트는 이동 객체의 ID(MOID)에 따라 메모리 상에 페이지(MOPage) 단위로 MORow의 MOTLocation을 저장하게 되는데, MOPage의 구조는 표 1과 같다. MOPage 구조에서 페이지 링크에는 데이타 화일 ID와 페이지 번호가 저장된다. 각 페이지는 고정된 길이를 갖기 때문에 메모리 상에 있는 페이지에 정해진 개수의 MOTLocation이 다 차게 되면 데이타 화일로 저장된다.

표 1. MOPage 구조

CurrentPageLink	현재 페이지 링크
PrevPageLink	같은 MOID를 갖는 이전 페이지 링크
NextPageLink	같은 MOID를 갖는 다음 페이지 링크
MOID	이동 객체의 ID
DataLength	페이지에 저장된 MOTLocation 개수
MBR	페이지에 저장된 데이타의 MBR
FromTime	가장 처음에 저장된 데이타의 시간
ToTime	가장 나중에 저장된 데이타의 시간
MOTLocations	DataLength개의 MOTLocation

3.1.2 디렉토리 구조

화일 기반 이동 객체 저장 컴포넌트는 그림 5와 같은 디렉토리 구조로 데이타베이스, 테이블, 메타데이타, 로그 등을 관리한다. Database 디렉토리 하위에는 데이타 화일과 인덱스 화일이 저장되는 Table 디렉토리, 메타데이타 화일이 저장되는 Metadata 디렉토리, 그리고 로그 화일이 저장되는 Log 디렉토리가 있다.

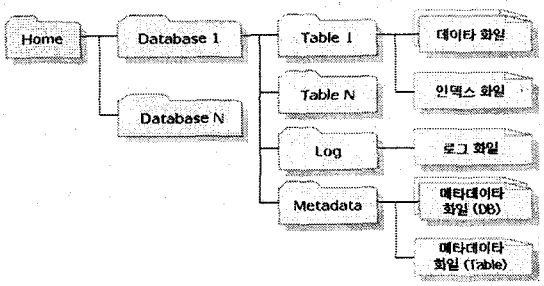

그림 5. 디렉토리 구조

3.2 다중 연결 관리자의 설계

다중 연결 관리자는 단순 질의 처리기와 인덱스 관리자의 인스턴스를 생성하며, 다중 사용자 연결을 지원하기 위해 DB 메타데이타에서 설정된 데이타베이스 사용

포트 개수만큼의 채널을 생성한다. 그리고, 클라이언트가 원격에서 IP와 Port 번호를 이용해서 화일 기반 이동 객체 저장 컴포넌트에 접속하게 되면, 다중 연결 관리자는 이용 가능한 단순 질의 처리기의 채널 정보를 클라이언트에게 넘겨줌으로써 클라이언트가 원격에서 질의를 요청할 수 있게 해준다.

3.3 단순 질의 처리기의 설계

클라이언트는 다중 연결 관리자를 통해 얻은 채널 정보를 이용하여 단순 질의 처리기에 접속한 후, 질의를 요청할 대상 테이블을 선택한다. 단순 질의 처리기는 클라이언트가 요청한 질의를 수행하기 위해 인덱스 관리자의 함수를 호출하며, 인덱스 관리자가 질의 처리 결과를 넘겨주면 그 결과를 클라이언트에게 전달한다.

3.4 인덱스 관리자의 설계

인덱스 관리자는 데이타베이스에 존재하는 각 테이블마다 하나의 인덱스 객체를 생성하고 관리한다. 또한, 인덱스 관리자는 메타데이타 관리자와 로그 관리자의 인스턴스를 생성하며 이를 관리하는 기능을 수행한다.

각 인덱스 객체는 C#에서 제공하는 HashTable을 이용하여 메모리 상에 MOID별로 하나의 페이지를 할당하여 관리하기 때문에 위치 데이타 저장 시 디스크 접근 횟수를 줄여주고, 검색 시에도 최근 데이타에 대한 빠른 검색을 제공한다. 그러므로, 단순 질의 처리기가 특정 테이블에 대해 저장 및 검색 질의 처리를 요청하면, 인덱스 관리자는 해당 테이블에 대해 저장과 검색을 수행하는 인덱스 객체에게 그 요청을 전달한다.

인덱스 객체는 저장 요청이 들어오면 먼저 메모리 상에 존재하는 해당 MOID의 페이지에 데이타를 저장하고, 해당 페이지가 다 차게 되면 데이타 화일 관리자에게 해당 페이지의 저장을 요청한다. 또한, 검색 요청이 들어오면 인덱스 객체는 메모리 상에서 검색을 수행하고, 인덱스에서 검색되지 않는 데이타가 있는 경우 데이타 화일 관리자에게 검색을 요청한 후, 그 결과를 취합한다. 그리고, 인덱스 관리자는 인덱스 객체가 수행 결과를 넘겨주면 로그 관리자를 통해 그 결과를 로그로 기록하고 단순 질의 처리기에게 전달한다.

3.5 데이타 화일 관리자의 설계

데이타 화일 관리자는 테이블을 생성하거나 삭제하고 데이타 화일을 생성하며 위치 데이타 저장 속도를 높이기 위해 디스크 공간을 미리 할당한 후에 페이지 단위로 저장하는 기능을 제공한다. 또한, MOID와 시간을 조건으로 하는 검색 및 삭제 질의를 수행한다. 데이타 화일 관리자에 의해 생성되는 데이타 화일은 Head Block과 Body Block으로 나뉘며 구조는 표 2와 같다.

표 2. 데이타 화일의 구조

Head Block	
HeaderSize	데이타 화일의 헤더 크기
Version	데이타 화일의 버전
FileID	데이타 화일의 ID
PageLength	화일에 저장된 페이지 개수
MaxDataLength	페이지 당 최대 MOTLocation의 개수
PageSize	페이지 크기
LastPageNumber	가장 마지막에 저장된 페이지 번호
LastDeletePage	가장 마지막에 삭제된 페이지 번호
MBR	화일에 저장된 모든 데이타의 MBR
FromTime	가장 처음에 저장된 데이타의 시간
ToTime	가장 나중에 저장된 데이타의 시간
Body Block	
MOPages	PageLength개의 MOPage

3.6 인덱스 화일 관리자의 설계

인덱스 화일 관리자는 인덱스 화일을 생성하고 이동 객체의 인덱스 정보를 저장, 검색, 삭제하는 기능을 담당한다. 인덱스 화일은 테이블마다 하나씩 생성되며, 해당 테이블의 데이타 화일에 저장된 모든 이동 객체에 대한 인덱스 정보가 인덱스 화일에 저장된다. 이동 객체의 인덱스 정보는 이동 객체의 MOID, 해당 MOID의 첫 번째 페이지 링크, 마지막 페이지 링크를 포함한다.

3.7 메타데이타 관리자의 설계

메타데이타 관리자는 데이타베이스에 대한 메타데이타 정보인 DB 메타데이타 정보를 관리하고 이를 화일에 저장하는 기능을 수행한다. DB 메타데이타 구조는 표 3과 같다.

표 3. DB 메타데이타의 구조

DBName	데이타베이스 이름
DBPath	데이타베이스 경로
DBProcessorLength	데이타베이스 프로세스 개수
DBProcessorPorts	데이타베이스 사용 포트 번호
DBPorts	데이타베이스 사용 포트 개수
LogFileName	데이타베이스 로그 화일 이름
LogOption	데이타베이스 로그 기록 옵션
LogBufferLength	데이타베이스 로그 버퍼 개수
StorageMBR	데이타베이스에 저장된 모든 데이타의 MBR
MOTLength	데이타베이스에 저장된 MOTLocation 개수
Shutdown	데이타베이스 정상 종료 여부

메타데이타 관리자는 테이블이 생성될 때마다 Table 메타데이타 화일을 생성하는데, 이 화일에는 해당 테이블 안에 생성된 모든 데이타 화일에 대한 메타데이타 정보가 저장된다. Table 메타데이타 구조는 표 4와 같고, 새로운 데이타 화일이 생성되면 그에 대한 메타데이타 정보가 추가된다.

표 4. Table 메타데이타의 구조

TableName	테이블 이름
FileID	데이타 화일 ID
FileName	데이타 화일 이름
MBR	데이타 화일에 저장된 모든 데이타의 MBR
FomeTime	데이타 화일에 가장 처음에 저장된 데이타 시간
ToTime	데이타 화일에 가장 나중에 저장된 데이타 시간
PageSize	데이타 화일의 페이지 크기
Update	데이타 화일에 대한 메타데이타 정보가 화일에 기록되었는지의 여부

3.8 로그 관리자의 설계

로그 관리자는 로그 화일을 생성하고, DB 메타데이타에서 설정된 로그 버퍼 개수만큼의 로그 버퍼를 생성한다. 또한, DB 메타데이타에서 설정된 로그 옵션에 따라 로그를 생성한 다음 이를 로그 버퍼에 저장하고, 로그 버퍼가 다 차면 버퍼에 있는 내용을 로그 화일에 기록하는 기능을 수행한다.

3.9 관리툴의 설계

관리툴은 화일 기반 이동 객체 저장 컴포넌트를 구동시키고, 데이타베이스의 생성, 삭제, 시작, 종료, 그리고 테이블의 생성 및 삭제 기능을 수행한다. 또한, 등록된 데이타베이스 목록을 검색하고, 각 데이타베이스의 테이블 목록, 로그 화일, 메타데이타 화일을 검색한다.

관리툴은 현재 하나의 데이타베이스만 구동시킬 수 있으며, 데이타베이스를 구동시키면 해당 데이타베이스에 대한 다중 연결 관리자가 생성됨으로써 클라이언트가 다중 연결 관리자와 접속할 수 있게 된다.

4. 시스템의 구현

본 장에서는 화일 기반 이동 객체 저장 컴포넌트의 구현에 대해 설명한다. 화일 기반 이동 객체 저장 컴포넌트를 구현하기 위하여 운영체제는 Microsoft Windows XP를 사용하였고, Microsoft .NET Framework v1.0.3705 상에서 개발 도구는 Microsoft Visual Studio .NET, 그리고 개발 언어는 C#을 사용하였다.

4.1 다중 연결 관리자의 구현

다중 연결 관리자는 그림 6과 같은 함수를 제공하는데, 먼저 Startup 함수가 호출되면 인덱스 관리자의 인스턴스가 생성된다. 그리고 나서, 클라이언트와의 연결을 위해 구현된 클래스인 MOConnectionAPI와 단순 질의 처리기를 구현한 MOSimpleQueryProcessor의 인스턴스를 생성한다. 그리고, LocalCMStart 함수를 호출하여 클라이언트가 접속할 수 있는 채널을 열고, LocalSQPStart 함수를 호출하여 DB 메타데이타에서 설정된 프로세스 개수만큼의 단순 질의 처리기를 구동시킨 후에 클라이언트의 연결을 기다린다.

```
public string Startup(string path, string name)
private int LocalCMStart(int port)
private int LocalSQPStart(int port)
```

그림 6. 연결 관리자의 함수

4.2 단순 질의 처리기의 구현

단순 질의 처리기에서 구현된 주요 함수는 그림 7과 같다. 단순 질의 처리기에서는 테이블을 생성하고 삭제하기 위한 CreateTable과 DeleteTable 함수를 제공하며, 이동 객체의 위치 데이타를 저장하고 삭제하기 위해 InsertRow와 Delete 함수를 제공한다. 또한, 위치 데이타의 검색을 위해 MOID를 기반으로 한 다양한 검색을 지원하는데, IDSearch 함수는 파라미터에 따라 모든 MOID에 대한 검색, 특정 MOID들에 대한 검색, 또는 특정 MOID들의 주어진 시간에 대한 검색 기능을 제공하는 함수이다. Search_PrevMOTLocation 함수와 Search_NextMOTLocation 함수는 특정 시점 이전 데이타와 이후 데이타를 주어진 개수만큼 검색한다.

```
public string CreateTable(string tableName)
public string DeleteTable(string tableName)
public bool InsertRow(MORow moRow)
public void Delete(MOID id)
public IEnumerator IDSearch()
public IEnumerator IDSearch(MOID[] ids)
public IEnumerator IDSearch(MOID[] ids, MOPeriod
period)
public IEnumerator Search_PrevMOTLocation(MOID id,
MOTime Time, int num)
public IEnumerator Search_NextMOTLocation(MOID id,
MOTime Time, int num)
```

그림 7. 단순 질의 처리기의 함수

4.3 인덱스 관리자의 구현

인덱스 관리자는 단순 질의 처리기의 요청에 따라 이동 객체의 위치 데이타에 대한 저장 및 검색 기능을 제공한다. 인덱스 관리자는 먼저 메타데이타 관리자를 생성하고, 메타데이타 관리자를 통해 로그 화일의 이름과 로그 기록 옵션 정보를 가져온 다음 로그 관리자를 생성한다. 그리고, 해당 데이타베이스의 테이블 개수만큼 인덱스 객체를 생성한다.

인덱스 관리자에서 제공하는 함수는 그림 8과 같으며, CreateTable, DeleteTable, InsertRow, 그리고 Delete 함수는 단순 질의 처리기에서 같은 이름으로 된 함수에 의해서 호출된다. SearchTable 함수는 주어진 테이블에 대한 인덱스 객체를 검색하며, SearchAllId 함수는 해당 테이블에 저장된 모든 이동 객체의 MOID를 검색한다. Search 함수는 특정 MOID의 주어진 시간에 대한 검색 기능을 제공하고 PrevMOTLocations와 NextMOTLocations 함수는 특정 시점 이전 데이타와 이후 데이타를 주어진 개수만큼 검색한다.

```
public string CreateTable(string tableName)
public string DeleteTable(string tableName)
public bool InsertRow(MORow moRow)
public void Delete(MOID id)
public MOHash SearchTable(string tableName)
public IEnumerator SearchAllId()
public MORow SearchAll(MOID moid)
public MORow Search(MOID moid,
MOTime fTime, MOTime tTime)
public MORow PrevMOTLocations(MOID moid,
MOTime Time, int num)
public MORow NextMOTLocations(MOID moid,
MOTime Time, int num)
```

그림 8. 인덱스 관리자의 함수

4.4 데이타 화일 관리자의 구현

데이타 화일 관리자는 그림 9와 같은 함수를 제공한다. CreateTable과 DropTable 함수는 테이블 생성과 삭제 기능을 수행하며, LocalCreateDataFile 함수는 데이타 화일을 생성한다. Write 함수는 이동 객체의 위치 데이타를 페이지 단위로 저장하며, 페이지 저장 요청이 들어오면 페이지 번호를 이용해 데이타 화일의 오프셋을 이동시키고 해당 페이지에 저장된 데이타를 화일에 기록한 다음, 화일 헤더 정보를 변경하고 해당 MOID에 대한 인덱스 정보를 추가하거나 변경하기 위해 LocalUpdateIndexFile 함수를 호출한다. 또한, 이동 객체의 위치 데이타에 대한 검색 요청이 들어오면 해당 MOID의 인덱스 정보를 이용해 필요한 페이지를 찾아 LocalReadPage 함수를 호출하여 페이지를 읽은 후 검색을 수행하고 그 결과를 반환한다. SearchAll 함수는 특정 MOID의 모든 데이타를 검색하며, SearchFirst 함수는 특정 MOID의 첫 번째 데이타를 검색한다. Search 함수는 해당 MOID의 주어진 시간에 속하는 데이타에 대한 검색을 수행한다. PreviousSearch와 NextSearch 함수는 해당 MOID의 특정 시점 이전 데이타와 이후 데이타를 검색한다. 그리고, Delete 함수는 주어진 MOID의 모든 데이타를 삭제하는 기능을 수행한다.

```
public int CreateTable(string tableName)
public int DropTable(string tableName)
private int LocalCreateDataFile()
public int Write(MOPage page)
private bool LocalUpdateIndexFile(MOPage page)
private MOPage LocalReadPage(PageLink pLink)
public MORow SearchAll(MOID moID)
public MOTLocation SearchFirst(MOID moID)
public MORow Search(MOID moID, MOTime fromTime,
MOTime toTime)
public MORow PreviousSearch(MOID moID, MOTime
fromTime, uint length)
public MORow NextSearch(MOID moID,
MOTime fromTime, uint length)
public int Delete(MOID moID)
```

그림 9. 데이타 화일 관리자의 함수

4.5 인덱스 화일 관리자의 구현

인덱스 화일 관리자는 테이블 별로 하나의 인덱스 화일을 생성하고 관리하는데, 인덱스 정보는 MOID를 키 값으로 가지고, 첫 번째 페이지 링크와 마지막 페이지 링크를 value 값으로 갖는 SortedList에 저장함으로써 인덱스 정보에 대한 빠른 저장과 검색을 제공한다. SortedList의 내용은 인덱스 화일 관리자가 소멸될 때 화일로 기록되며, 생성될 때 화일로부터 인덱스 정보를 읽어온다.

인덱스 화일 관리자는 그림 10과 같은 함수를 제공한다. AddIndex, UpdateIndex, DeleteIndex 함수는 각각 해당 MOID의 인덱스 정보를 추가하고 변경하거나 삭제한다. SearchFirstPage 함수는 해당 MOID의 첫 번째 페이지 링크를 검색하며, SearchLastPage 함수는 해당 MOID의 마지막 페이지 링크를 검색한다.

```
public bool AddIndex(MOID moID,
PageLink firstPage)
public bool UpdateIndex(MOID moID, PageLink
lastPage)
public bool DeleteIndex(MOID moID)
public PageLink SearchFirstPage(MOID moID)
public PageLink SearchLastPage(MOID moID)
```

그림 10. 인덱스 화일 관리자의 함수

4.6 메타데이타 관리자의 구현

메타데이타 관리자는 DB 메타데이타 화일과 Table 메타데이타 화일을 생성하고 관리하며, 모든 메타데이타 화일은 xml 화일로 생성된다. 메타데이타 관리자에서 제공하는 기능은 그림 11에 있는 함수로 구현되었다. WriteDBMetadata와 LocalReadDBMetadata 함수는 DB 메타데이타 화일을 읽고 쓰기 위한 함수이다. Add 함수는 새로운 데이타 화일이 생성된 경우 그에 대한 정보를 Table 메타데이타 정보에 추가한다. Update 함수는 데이타 화일의 헤더가 변경된 경우 그 내용을 Table 메타데이타 정보에 반영한다.

```
public int WriteDBMetadata()
private int LocalReadDBMetadata()
public long Add(string tableName, long fileID, string
fileName, int pageSize)
public int Update(string tableName, long fileID, MOMBR
mbr, MOTime from, MOTime to)
```

그림 11. 메타데이타 관리자의 함수

4.7 로그 관리자의 구현

로그 관리자는 그림 12에 있는 함수를 제공하는데, AddLog 함수는 로그 버퍼에 로그를 추가하며, LocalWrite 함수는 로그 화일에 로그를 기록한다. ChangeLogBuffer 함수는 로그 버퍼를 교환하고, ChangeLogFile 함수은 로그 화일을 새로운 로그 화일을 생성하는 기능을 수행한다.

```
private int LocalWrite(int logBufferNumber)
public int AddLog(string query, bool success, MORow
morow, string err)
public int ChangeLogBuffer()
public string ChangeLogFile()
```

그림 12. 메타데이타 관리자의 함수

4.8 관리툴의 구현

관리툴은 화일 기반 이동 객체 저장 컴포넌트를 구동시키며, 데이타베이스 목록과 데이타베이스의 테이블 목록, 메타데이타 정보, 로그 화일 목록을 볼 수 있다. 또한, 데이타베이스를 새로 생성하거나 삭제할 수 있고, 테이블의 생성 및 삭제도 가능하다. 관리툴에서는 하나의 데이타베이스만 구동시킬 수 있으며, 관리툴의 인터페이스는 그림 13과 같다.

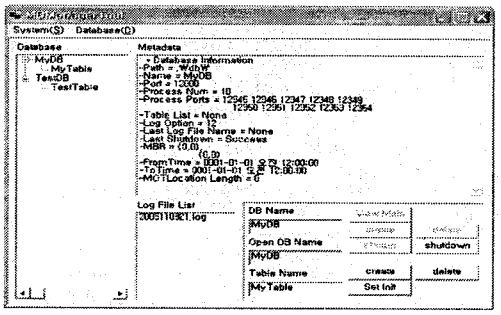

그림 13. 관리툴의 인터페이스

4.9 클라이언트의 구현

클라이언트의 인터페이스는 그림 14와 같으며, 클라이언트에서는 화일 기반 이동 객체 저장 컴포넌트에 접속해서 테이블을 생성하거나 삭제할 수 있고, 임의로 생성된 이동 객체의 위치 데이타를 저장할 수 있다. 또한, 저장되어 있는 데이타를 검색하거나 삭제하는 기능도 제공된다. 화일 기반 이동 객체 저장 컴포넌트는 다중 사용자를 지원하므로 동시에 여러 클라이언트에서 저장 및 검색 질의를 요청할 수 있다.

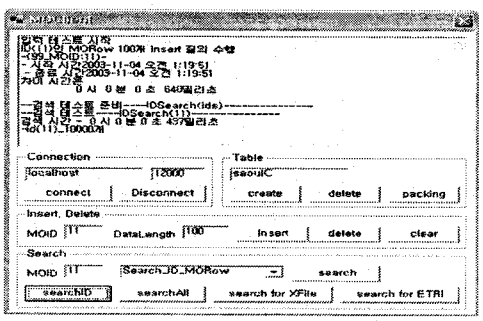

그림 14. 클라이언트의 인터페이스

5. 결론 및 향후 연구

최근 이동 객체의 위치정보를 활용한 다양한 위치 기반 서비스가 가능해짐에 따라 위치정보의 저장, 검색, 갱신을 위한 이동 객체 데이타베이스에 대한 필요성이 증가하고 있다. 이동 객체의 궤적을 보존하는 이동 객체 데이타베이스에서 기존의 상용 데이타베이스 시스템을 저장 시스템으로 사용할 경우 저장 공간을 많이 차지하고, 대용량의 위치 데이타에 대한 불필요한 트랜잭션 연산으로 인한 큰 오버헤드가 발생한다는 단점이 있다.

이에 본 논문에서는 대용량의 위치 데이타를 보다 효율적으로 처리하고 관리할 수 있는 화일 기반 이동 객체 저장 컴포넌트를 설계 및 구현하였다. 화일 기반 이동 객체 저장 컴포넌트는 원격에 있는 다중 사용자 관리를 위한 다중 연결 관리자, 저장 및 검색 질의를 처리하는 단순 질의 처리기, 최근 데이타에 대한 빠른 저장과 검색을 제공하는 인덱스 관리자, 메타데이타 정보를 관리하는 메타데이타 관리자, 이동 객체의 위치 데이타를 데이타 화일에 저장하고 검색하는 데이타 화일 관리자, 이동 객체에 대한 인덱스 정보를 관리하는 인덱스 화일 관리자, 저장과 검색에 대한 질의 수행 결과를 로그로 남기기 위한 로그 관리자, 화일 기반 이동 객체 저장 컴포넌트를 구동시키기 위한 관리툴, 그리고 테스트를 위한 클라이언트로 구성된다.

향후 연구 과제는 OLE DB 데이타 제공자를 추가하는 것과, 데이타 화일에 대한 압축 기능을 제공함으로써 저장 공간을 보다 효율적으로 사용할 수 있도록 하는 것이다.

참고문헌

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

[2] 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, 2000, 25(1): pp.1-42.

[3] Obermeyer, P., and Hawkins, J., *Microsoft .NET Remoting: A Technical Overview,* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/introremoting.asp, 2001.

[4] Sistla, A.P., Wolfson, O., Chamberlain, S., and Dao, S., "Modeling and Querying Moving Objects," ICDE, 1997, pp.422-432.

[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] Wolfson, O., Xu, B., Chamberlain, S., and Jiang, L., "Moving Objects Databases: Issues and Solutions," SSDBM, 1998, pp.111-122.

[7] 장유정, 윤재관, 한기준, "위치관리 컴포넌트를 위한 분산 위치 저장 컴포넌트," 한국정보과학회 학술발표논문집, 30권2호, 2003, pp.226-228.

[8] 조대수, 남광우, 이재호, 민경욱, 장인성, 박종현, "대용량 위치 데이타 관리를 위한 정보 시스템," 한국정보과학회 데이타베이스 연구, 18권4호, 2002, pp.11-21.