유비쿼터스 환경을 위한 시공간 미들웨어의 설계 및 구현

(Design and Implementation of a Spatio-Temporal Middleware for Ubiquitous Environments)

김 정 준* 정 연 종** 김 동 오*** 한 기 준****

(Jeong-Joon Kim)(Yeon-Jong Jeong) (Dong-Oh Kim) (Ki-Joon Han)

요 약 최근 인간 중심의 미래 컴퓨터 환경인 유비쿼터스 컴퓨팅 환경 실현을 위한 기술의 연구 개발이 활발히 진행됨에 따라 시공간 데이타를 다루는 GIS 기술이 대두되고 있다. 또한, 다양한 이기종 환경에서 호환성을 제공하기 위해서 생성된 데이타를 가공하여 사용자에게 원만하게 서비스를 제공하기 위한 미들웨어의 필요성이 증가되고 있다. 이와 같은 미들웨어의 핵심 기술로는 위치 측위 시스템으로부터 끊임없이 연속으로 들어오는 데이타 스트림을 실시간으로 처리하는 기술과 다양한 컴퓨팅 디바이스간의 데이타 동기화를 위한 기술이 요구된다. 하지만, 기존의 데이타 스트림 처리 기술은 비시공간 데이타를 대상으로 개발되었기 때문에 시공간 데이타에 대한 질의 처리가 효율적으로 지원되지 않는 문제점이 있다. 따라서, 본 논문에서는 모바일 시공간 DBMS(DataBase Management System)와 서버 시공간MMDBMS (Main Memory DBMS)간의 상호운용성을 제공하기 위한 시공간 미들웨어를 설계 및 구현하였다. 시공간 미들웨어는 끊임없이 연속으로 들어오는 시공간 데이타 스트림을 실시간으로 처리하고 시공간 DBMS간의 양방향 동기화를 지원하여 이기종 장치들간의 호환성을 유지하고 질의 처리를 하는데 있어 데이타의 무결성을 보장한다. 또한, 각 시공간 DBMS와 연결된 세션의 상태를 관리하고, 시공간 미들웨어가 안정적으로 작동될 수 있기 위한 자원 관리 기능을 제공한다. 마지막으로, 본 논문에서는 시공간 미들웨어를 실시간 위치 추적 시스템에 적용해 봄으로써 시공간 미들웨어의 효용성을 입증하였다.

키워드 : 시공간 미들웨어, 시공간 데이타, 시공간 데이타 스트림, 양향방 동기화

Abstract As R&D(Research and Development) is going on actively to develop technologies for the ubiquitous computing environment, which Is the human-oriented future computing environment, GIS dealing with spatio-temporal data is emerging as a promising technology. This also increases the necessity of the middleware for providing services to give interoperability in various heterogeneous environments. The core technologies of the middleware are real-time processing technology of data streams coming unceasingly from positioning systems and data stream processing technology developed for non-spatio-temporal data. However, it has problems in processing queries on spatio-temporal data efficiently. Accordingly, this paper designed and implemented the spatio-temporal middleware that provides interoperability between a mobile spatio-temporal DBMS(DataBase Management System) and a server spatio-temporal MMDBMS(Main Memory DataBase Management System). The spatio-temporal middleware maintains interoperability among heterogeneous devices and guarantees data integrity in query processing through real-time processing of unceasing spatio-temporal data streams and two way synchronization of spatio-temporal DBMSs. In addition, it manages session for the connection of each spatio-temporal DBMS and manages resources for its stable operation. Finally, this paper proved the usability of the spatio-temporal middleware by applying it to a real-time position tracking system.

Keywords : Spatio-temporal Middleware, Spatio-temporal Data, Spatio-temporal Data Stream, Two-way Synchronization

† 본 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.

* 건국대학교 컴퓨터공학과 박사과정, jjkim9@db.konkuk.ac.kr

** 건국대학교 컴퓨터공학과 석사, yjjeong@db.konkuk.ac.kr

*** 건국대학교 컴퓨터공학과 강의교수, dokim@db.konkuk.ac.kr(교신저자)

**** 건국대학교 컴퓨터공학과 교수, kjhan@db.konkuk.ac.kr

논문접수 : 2009.02.13 수정일 : 1차 2009.03.14 심사완료 : 2009.03.16

1. 서 론

최근 5C(Computing, Communication, Connectivity, Contents, Calm) 또는 5Any(Any-time, Any-where, Any-network, Any-device, Any-service)를 지향하는, 즉 모든 장소에 컴퓨터가 있어 그것을 자유롭게 누구나 사용할 수 있는 인간 중심의 미래 컴퓨터 환경인 유비쿼터스 컴퓨팅 환경(Ubiquitous Computing Environments) 실현을 위한 기술의 연구 개발이 활발히 진행되고 있다[1,2].

또한, RFID(Radio Frequency IDentification)나 GPS (Global Positioning System) 등과 같은 위치 측위 기술의 발달로 유비쿼터스 컴퓨팅 환경에서 시공간 데이타를 다루는 GIS 기술이 미래의 핵심 기술로 대두되고 있으며, 이기종 환경에서 다양한 컴퓨팅 디바이스가 생성하는 대량의 정보를 가공하여 사용자 또는 응용 서비스에게 전달해주는 미들웨어 기술이 요구되고 있다[3,4,5]. 이와 같은 미들웨어의 핵심 요소 기술로 위치 측위 시스템으로부터 끊임없이 연속으로 들어오는 데이타 스트림을 실시간으로 처리하고 다양한 컴퓨팅 디바이스간의 데이타 동기화를 위한 기술을 들 수 있다. 하지만, 기존의 데이타 스트림 처리 기술은 비시공간 데이타를 대상으로 개발되었기 때문에 시공간 데이타에 대한 질의 처리가 효율적으로 지원되지 않는 문제점이 있다. 따라서, 시공간 데이타를 다루는 미들웨어의 연구 개발 실정은 미흡한 상태이다[6,7,8,9].

따라서 본 논문에서는 모바일 시공간 DBMS와 서버 시공간 MMDBMS간의 상호운용성을 제공하기 위한 시공간 미들웨어를 설계 및 구현하였다. 시공간 미들웨어는 시공간 데이타 스트림을 실시간으로 처리하고 관리하기 위해 스탠포드 대학에서 연구 중이며 소스가 공개되어 있는 STREAM(STanford stREam dAta Manager)을 기반으로 시공간 질의 처리 기능을 지원하도록 확장하여 사용하였다[10]. 또한, OGC(Open Geospatial Consortium)에서 제시한 “Simple Feature Specification for SQL”을 확장한 시공간 데이타 타입을 지원한다[11].

시공간 미들웨어는 모바일 시공간 DBMS와 서버 시공간 MMDBMS간의 양방향 동기화를 지원함으로써 이기종 장치들간의 호환성을 보장하고, 각 시공간 DBMS와 접속된 세션의 상태를 관리하여 메모리를 효율적으로 사용하며, 시공간 미들웨어가 안정적으로 작동될 수 있기 위한 자원 관리 기능을 제공한다. 마지막으로, 시공간 미들웨어를 유비쿼터스 환경에서 각광받고 있는 실시간 위치 추적 시스템에 시나리오를 설정하여 적용해 봄으로써 미들웨어의 효용성을 실험하였다.

본 논문의 구성은 다음과 같다. 제2장에서 관련 연구로 스탠포드 대학에서 개발한 DSMS(Data Stream Management System)인 STREAM 프로젝트를 분석하고, OGC에서 제시한 “Simple Feature Specification for SQL”에 대해 언급한다. 제3장에서는 시공간 미들웨어의 전체 구조와 상세 설계에 대하여 설명한다. 제4장에서는 설계에 따라 구현된 내용에 대해 언급하고, 실시간 관리가 필요한 시나리오를 통해서 시공간 미들웨어의 효용성을 실험한다. 마지막으로, 제5장에서는 결론에 대하여 기술한다.

2. 관련 연구

본 장에서는 본 논문에서 확장하여 사용한 대표적인 DSMS인 STREAM에 대하여 분석하며, OGC에서 제공한 Simple Feature Specification for SQL에 대하여 설명한다.

2.1 STREAM

STREAM은 데이타 스트림을 모니터링하고 필터링하기 위하여 Stanford 대학에서 개발한 DSMS이며 현재 서버 0.6 버전과 클라이언트 0.3 버전이 소스와 함께 BSD License 형태로 공개되어 있다[10]. 서버는 Linux 기반에서 C++로 구현되어 있으며 라이브러리로 사용이 가능하고, 데이타 스트림 처리를 위한 연속 질의 언어 CQL(Continuos Query Language)을 지원한다. 클라이언트는 운영체제에 독립적인 Java로 구현되어 있으며 데이타 및 질의 입력, 실행 계획 뷰어 등을 제공한다. 그림 1은 STREAM의 동작 방식을 보여주고 있다.

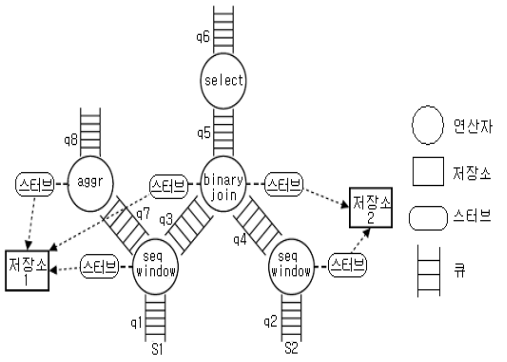


그림 1. STREAM의 동작 방식

사용자가 등록한 연속 질의는 Select, Project, Join 등의 연산자가 트리 형태로 모여 있는 실행 계획으로 변환되어 저장 및 처리된다. 각 연산자는 필요에 따라 저장소를 생성하게 되며 이 저장소에는 질의 처리의 중간 결과가 저장되어 여러 연산자에 의해 공유된다. 각 연산자는 스터브(Stub)라는 상태 저장 모듈을 갖고 있어서 저장소에 대한 참조 값을 보관하며, 입력 큐의 데이타들을 처리하여 출력 큐에 저장한다. 이 외에도, 하나의 질의 결과가 다른 질의들에 의해 공유되기 때문에 질의 처리 속도 및 공간 효율성이 향상된다.

STREAM은 CQL이라는 자체적으로 정의한 연속 질

의 언어를 사용한다[12]. CQL은 데이타 스트림으로부터 입력된 데이타를 릴레이션으로 바꾼 뒤 실행 계획에 따라 데이타를 처리하고, 결과를 스트림이나 릴레이션으로 출력한다. 여기서 스트림이란 갱신 및 삭제는 발생하지 않고 지속적인 입력만 발생하는 모델이며, 릴레이션이란 입력과 삭제가 발생하며 크기가 가변적이긴 하지만 논리적으로 유한한 모델이다. 데이타 스트림은 무선 센서 네트워크나 웹 서버 등으로부터 무한하게 입력되기 때문에 모든 입력 데이타에 대하여 질의를 처리한 후에 처리 결과를 돌려주려고 할 경우 실행 시간이 무한대가 될 수밖에 없다. 이 때문에 CQL에서는 슬라이딩 윈도우(Sliding Window) 기법을 사용하여 무한한 스트림을 유한한 릴레이션으로 변환한다. 본 논문에서는 이러한 STREAM을 확장하여 시공간 DSMS를 개발하기 위해 OGC에서 제시한 “Simple Feature Specification for SQL” 표준 명세에 따라 공간 데이타 타입 및 연산자를 확장하였다.

2.2 OGC SQL 스키마

OGC에서 제안한 SQL 스키마인 “Simple Feature Specification for SQL”은 ODBC API를 경유하는 심플 공간 피쳐 집합의 저장, 검색, 질의, 그리고 갱신을 지원하는 표준 SQL 스키마를 정의하고 있다[11]. 심플 피쳐는 공간과 비공간 속성을 모두 지원하기 위해 OGC의 “OpenGIS 추상 명세(OpenGIS Abstract Specification)”에 의해 정의되어 있다. 공간 속성은 Geometry 값이며, Simple Feature는 점 사이의 선형 보간법을 이용한 2D Geometry를 기반으로 한다. “Simple Feature Specification for SQL”에서 OpenGIS Geometry 모델을 기반으로 하는 표준 SQL Geometry 타입은 OpenGIS Geometry 모델을 기반으로 하여 계층 구조로 구성되어 있다.

계층 구조의 최상위 타입인 Geometry는 Point, Curve, Surface, GeometryCollection의 하위 타입을 갖는다. 여기서 GeometryCollection은 이종의(Heterogeneous) Geometry들의 집합이다. GeometryCollection의 하위 타입인 MultiPoint, MultiCurve, MultiSurface는 동종의(Homogenous) Point, Curve, Surface들의 집합을 다루기 위해 사용된다. 차원을 살펴보면, 0차원의 Geometry 타입은 Point와 MultiPoint이고, 1차원의 Geometry 타입은 Curve와 MultiCurve, 그리고 이들 타입의 하위 타입들이다. 마지막으로 Surface와 MultiSurface, 그리고 이들 타입의 하위 타입들은 2차원의 Geometry 타입이다. 계층 구조에서 보이는 12개의 SQL Geometry 타입 중 Geometry, Curve, Surface, MultiCurve, MultiSurface는 인스턴스를 생성할 수 없는(Non-instantiable) 공간 데이타 타입이다.

3. 시스템 설계

본 장에서는 시공간 미들웨어의 전체 구조 및 상세 설계에 대해서 자세히 설명한다.

3.1 시스템 전체 구조

그림 2는 본 논문에서 개발한 유비쿼터스 환경을 위한 시공간 미들웨어의 전체 구조를 보여준다.

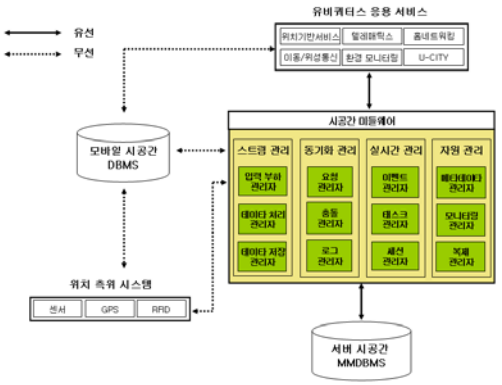


그림 2. 시스템 전체 구조

그림 2를 살펴보면 시공간 미들웨어는 유비쿼터스 컴퓨팅 환경을 위한 시스템으로서 센서, GPS, RFID와 같은 다양한 위치 측위 시스템으로부터 들어오는 시공간 데이타 스트림을 처리하고, 모바일 시공간 DBMS와 서버 시공간 MMDBMS와의 상호연계를 통해 시공간 데이타를 관리한다. 또한, 각 시공간 DBMS간의 동기화를 통해 위치 기반 서비스, 텔레매틱스, 홈네트워킹 등의 유비쿼터스 환경에서 다양한 지원 도구로써 활용이 가능하다.

시공간 미들웨어는 크게 스트림 관리, 동기화 관리, 실시간 관리, 자원 관리로 나누어진다. 스트림 관리는 데이타 스트림의 입력 부하를 줄이기 위한 입력 부하 관리자, 연속 질의를 지원하고 시공간 데이타 타입과 질의 내에 포함된 시공간 함수를 처리하여 질의 결과를 반환하는 데이타 처리 관리자, 데이타 스트림을 서버 시공간 MMDBMS에 저장하는 데이타 저장 관리자로 구성된다. 동기화 관리는 동기화 요청을 받아 동기화를 수행하는 요청 관리자, 동기화 수행 도중 데이타간의 충돌이 발생하면 이를 해결하는 충돌 관리자, 동기화를 위한 로그 정보를 관리하는 로그 관리자로 구성된다. 실시간 관리는 이벤트를 발생시키고 이벤트간의 연관관계를 관리하는 이벤트 관리자, 특정 이벤트 발생시 이를 처리하고 태스크를 효율적으로 관리하는 태스크 관리자, 각 시공간 DBMS와 접속된 세션의 상태를 관리하는 세션 관리자로 구성된다. 마지막으로 자원 관리는 시공간 미들웨어의 다수의 메타 데이타를 관리하는 메타데이타 관리자, 시공간 미들웨어와 서버 시공간 MMDBMS의 상태를 주기적으로 감시하는 모니터링 관리자, 서버 시공간 MMDBMS의 복제된 데이타를 저장하여 관리하는 복제 관리자로 구성된다.

3.2 스트림 관리

스트림 관리는 입력 부하 관리자, 데이타 처리 관리자, 데이타 저장 관리자로 구성된다.

3.2.1 입력 부하 관리자

입력 부하 관리자는 처리하지 못할 과도한 입력이 들어왔을 때 정확도 손실을 최소화 하면서 입력 데이타 스트림의 양을 줄여 과부하 문제를 해결하기 위한 기능을 제공한다. 이를 위해 시공간 객체를 여러 저장소에서 공유할 수 있도록 시공간 객체 아이디와 참조 카운터를 관리함으로써 시공간 객체의 처리 효율성을 향상시킨다. 그림 3은 시공간 객체가 여러 저장소에서 공유되는 예를 보여준다.

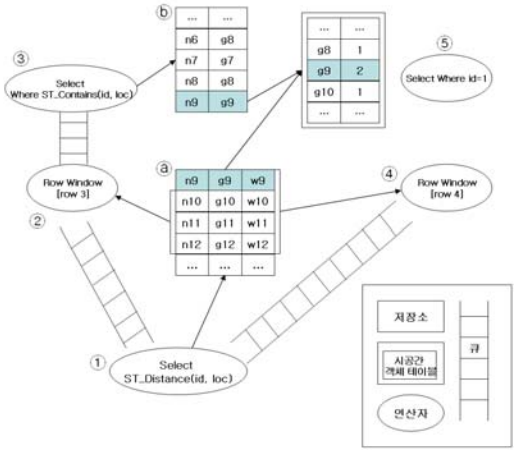


그림 3. 시공간 객체 공유 예

그림 3을 살펴보면 연산자 1이 ST_Distance 함수를 통해 시공간 객체를 생성한 뒤 이것을 저장소 a에 저장을 해두면 연산자 2와 4가 공유하게 된다. 이때, 연산자 2는 저장소 a에 있는 9번째 튜플을 처리하고 슬라이딩 윈도우를 10부터 12번 튜플로 옮긴 뒤 연산자 3이 이를 저장소 b에 저장하지만, 연산자 4가 9번째 튜플을 아직 처리하지 않아서 저장소 a에는 9번째 튜플이 남아있게 된다. 이러한 경우 저장소 a와 저장소 b 모두에 시공간 객체 g9가 존재하게 되며 g9에 대한 참조 카운터를 2로 늘린다. 연산자 4가 9번째 튜플을 처리하면 저장소 a에서 9번째 튜플이 삭제되는데, 이때 입력 부하 관리자는 Heap 영역에 있는 시공간 객체를 바로 삭제하지 않고 참조 카운터만 1로 줄인 후, 저장소 b에서도 9번째 튜플이 삭제되면 참조 카운터가 0이 되면서 g9를 삭제한다.

또한, 입력 부하 관리자는 데이타 스트림의 입력 부하 분산을 위해 필터링 기능을 제공한다. 그림 4는 데이타 스트림이 필터링되는 과정을 보여준다.

시공간 미들웨어에서는 필터링을 위해 좌표 거리 차이, 특정 이동 객체의 아이디 등이 필터링 조건으로 지원된다.

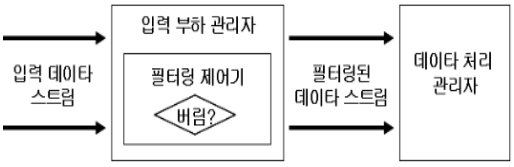


그림 4. 데이타 스트림의 필터링 과정

3.2.2 데이타 처리 관리자

데이타 처리 관리자는 등록된 질의를 분석하여 실행 계획을 생성하고 연속 질의를 처리하는 기능을 수행하며, 질의에 시공간 함수가 포함되어 있을 경우 이를 처리한다. 그리고 실행 계획 생성 시에 필요한 메모리 공간을 미리 확보하여 작업에 대한 최적화를 수행한다.

시공간 미들웨어에서 입력 질의에서 사용될 수 있는 시공간 데이타 타입과 시공간 함수로는 “Simple Feature Specification for SQL”에 제시된 데이타 타입과 공간 함수를 확장하여 사용하였다. 표 1은 시공간 미들웨어에서 지원하는 시공간 데이타 타입을 보여준다.

표 1. 시공간 데이타 타입

형 식	설 명	타 입
ST_POINT	시공간 점 객체를 표현한다.	double
ST_LINESTRING	시공간 선 객체를 표현하며, 각 선들은 여러개의 점으로 구성된다.	double
ST_POLYGON	시공간 다각형 객체를 표현하며, 각 다각형들은 여러개의 점으로 구성된다.	double
ST_MULTIPOINT	다중 시공간 점 객체를 표현한다.	double
ST_MULTILIN-ESTRING	다중 시공간 선 객체를 표현한다.	double
ST_MULTIPOLYGON	다중 시공간 다각형 객체를 표현한다.	double

표 1에서 나타내는 시공간 데이타 타입들을 지원하기 위하여 Geometry 타입을 설계하였다. Geometry 타입은 GEOS(Geometry Engine Open Source)가 생성하는 시공간 객체에 대한 포인터이다. GEOS는 공간 객체 모델과 기본적인 기하(geometric) 함수를 제공하기 위해 JAVA 언어로 작성된 API인 JTS(Java Topology Suite)를 C++ 언어로 포팅한 공개 소스이다[13]. 표 2는 시공간 미들웨어에서 지원하는 시공간 함수에 대한 설명을 보여준다.

표 2를 살펴보면 시공간 미들웨어는 시공간 관계 함수 9개와 시공간 분석 함수 5개를 지원한다.

표 2. 시공간 함수

종 류	이 름	설 명
시공간 관계 함수	ST_Equals	각 단위 시간 마다 두 시공간 객체가 동일하면 참을 반환한다.
	ST_Disjoint	각 단위 시간 마다 두 시공간 객체가 교집합이 없으면 참을 반환한다.
	ST_Touches	각 단위 시간 마다 두 시공간 객체의 경계선만 닿았으면 참을 반환한다.
	ST_Within	각 단위 시간 마다 첫 번째 시공간 객체가 두 번째 시공간 객체를 포함하면 참을 반환한다.
	ST_Overlaps	각 단위 시간 마다 두 시공간 객체가 같은 차원이면서 겹쳐 있으면 참을 반환한다.
	ST_Crosses	각 단위 시간 마다 두 시공간 객체가 교차하면서 교차된 영역의 차원이 같거나 줄어들면 참을 반환한다.
	ST_Intersects	각 단위 시간 마다 두 시공간 객체가 교차하면 참을 반환한다.
	ST_Contains	각 단위 시간 마다 두 번째 시공간 객체가 첫 번째 시공간 객체를 포함하면 참을 반환한다.
	ST_Relate	각 단위 시간 마다 두 시공간 객체의 위상관계가 주어진 조건(9-Intersection Matrix)을 만족하는지 여부 반환한다.
시공간 분석 함수	ST_Distance	두 시공간 객체간의 거리 변화를 시간 순으로 추출하여 반환한다.
	ST_Intersection	두 시공간 객체가 겹치는 부분을 시간 순으로 추출하여 반환한다.
	ST_Difference	첫 번째 시공간 객체에서 두 번째 시공간 객체를 뺀 부분을 시간 순으로 추출하여 반환함.
	ST_Union	두 시공간 객체를 합집한 시공간 객체를 시간 순으로 추출하여 반환한다.
	ST_Buffer	시공간 객체 주변을 주어진 크기만큼 버퍼링 해서 시간 순으로 계산, 추출하여 반환한다.

3.2.3 데이타 저장 관리자

데이타 저장 관리자는 위치 측위 시스템에서 입력되는 데이타 스트림을 서버 시공간 MMDBMS에 저장하는 기능을 제공한다. 데이타 스트림은 실시간으로 처리되기 때문에 DSMS와 같은 시스템에서는 데이타 스트림의 저장이 이루어지지 않지만, 위치 추적 서비스에서 이동체의 위치 동선을 통해 이동경로를 파악하는 경우 과거의 이력 데이타가 필요하다. 따라서, 데이타 저장 관리자는 데이타 스트림을 서버 시공간 MMDBMS에 저장하여 관리할 수 있는 기능을 제공하며, 데이타 스트림은 표준 인터페이스(ODBC)를 통해 서버 시공간 MMDBMS에 저장된다.

3.3 동기화 관리

동기화 관리는 요청 관리자, 충돌 관리자, 로그 관리자로 구성된다.

3.3.1 요청 관리자

요청 관리자는 데이타 동기화를 위하여 모바일 시공간 DBMS와 서버 시공간 MMDBMS로부터 들어오는 다수의 동기화 요청을 받아 동기화를 수행하는 기능을 제공한다. 동기화 방식은 모바일 시공간 DBMS 우선 방식, 서버 시공간 MMDBMS 우선 방식, 최근 데이타 우선 방식을 지원하는데, 각각은 최근 동기화 시점을 비교하여 정상적인 동기화 수행 가능 여부와 동기화를 수행할 시점을 판단하기 위해 각 로그 정보를 확인한다. 그림 5는 모바일 시공간 DBMS와 서버 시공간 MMDBMS간의 동기화가 진행되는 프로토콜을 보여준다.

그림 5를 살펴보면 동기화가 진행되는 일련의 과정은 다음과 같다. 먼저, 모바일 시공간 DBMS에서 동기화를 요청하면 시공간 미들웨어는 동기화 상태 정보 로그를 통해 해당 시공간 MMDBMS의 동기화 수행 여부를 확인한다. 동기화 수행이 가능하면, 동기화 완료 정보 로그를 통해 동기화를 수행할 각 시공간 DBMS의 최근 동기화 완료 시간을 확인한 후 각 시공간 DBMS들로부터 트랜잭션 정보 로그를 전송받아 비교한다. 이후, 각 시공간 DBMS에 반영할 질의를 생성하여 전송하고, 동기화 완료 정보 로그에 진행된 동기화 정보를 기록한다. 무분별한 로그 저장을 막기 위해 해당 시공간 DBMS에 갱신될 트랜잭션에 대한 로그 정보와 삭제될 로그 정보 시점을 전송하고, 마지막으로 동기화 완료 후 동기화 상태 정보 로그에 해당 시공간 DBMS의 동기화 상태 정보를 동기화 수행 가능 상태로 변경함으로써 동기화 과정이 끝나게 된다.

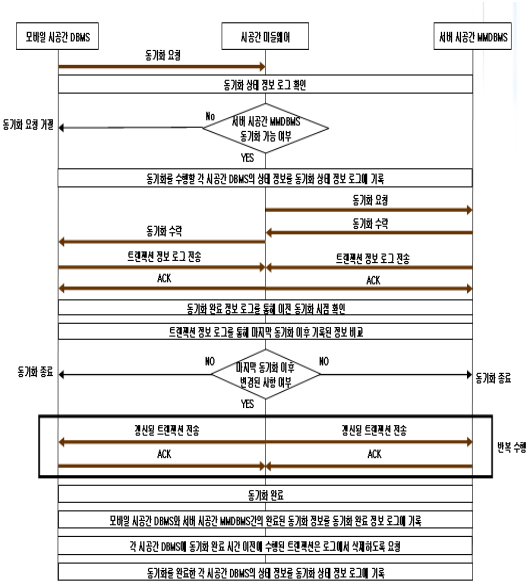


그림 5. 동기화 프로토콜

3.3.2 충돌 관리자

충돌 관리자는 동기화 수행시 데이타의 충돌 여부를 확인하고 해결하는 기능을 제공하고, 데이타 충돌 발생시 정해진 정책에 따라 데이타의 충돌 종류를 확인하고 이를 처리한다. 표 3은 데이타 충돌 발생 상황에 따른 해결 정책을 보여준다.

표 3에서 보는 바와 같이 최근 데이타 우선 방식으로 동기화 수행하는 도중 Duplication 충돌 발생시에는 트랜잭션 수행 시간도 동일하여 데이타의 우선순위를 판단할 수 없기 때문에 서버 시공간 MMDBMS를 우선시하여 서버 시공간 MMDBMS 우선 방식으로 해결한다. 그리고 Insert, Delete, Update 충돌 발생시에는 요청된 동기화 수행 방식에 따라 데이타의 우선순위를 판단하여 이를 해결하게 된다.

표 3. 충돌 해결 정책

데이타 충돌 종류	충돌 발생 상황	해결 정책
Duplication 충돌	최근 데이타 우선 방식으로 동기화시 동일한 고유 ID, 시간 값, 트랜잭션 수행 시간을 가진 데이타가 존재할 경우	서버 시공간 MMDBMS 우선 방식
Insert 충돌	동일한 고유 ID, 시간 값을 가진 데이타를 삽입할 때 이미 다른 값으로 삽입된 경우	모바일 시공간 MMDBMS 우선 방식
Delete 충돌	동일한 고유 ID, 시간 값을 가진 데이타가 한쪽만 삭제된 경우	서버 시공간 MMDBMS 우선 방식
Update 충돌	동일한 고유 ID, 시간 값을 가진 데이타가 다른 값으로 변경된 경우	최근 데이터 우선 방식

3.3.2 로그 관리자

로그 관리자는 동기화를 수행할 수 있는지의 상태 정보와 동기화 수행 완료시 해당 정보를 로그에 기록하고 관리하는 기능을 제공한다. 시공간 DBMS의 동기화 재요청시에는 로그를 이용하여 마지막 동기화 이후에 변화된 부분만 갱신하는 기법을 사용하기 위해 로그 관리자는 동기화를 수행할 수 있는지 여부의 정보를 저장하는 동기화 상태 정보 로그, 동기화 완료시 동기화를 수행한 각 시공간 DBMS의 정보를 저장하는 동기화 완료 정보 로그, 각 시공간 DBMS에서 수행된 트랜잭션 정보를 저장하는 트랜잭션 정보 로그를 관리한다.

3.4 실시간 관리

실시간 관리는 이벤트 관리자, 태스크 관리자, 세션 관리자로 구성된다.

3.4.1 이벤트 관리자

이벤트 관리자는 태스크의 수행 요청 및 종료시 이에 대한 이벤트를 발생시키고 태스크가 수행될 수 있도록 해당 관리자에게 전달하는 기능을 제공한다. 발생되는 이벤트의 종류로는 필터링 요청/종료 이벤트, 데이타 스트림 저장 요청/종료 이벤트, 데이타 스트림 시공간 MMDBMS 저장 요청/종료 이벤트, 각 시공간 DBMS간의 동기화 수행 요청/종료 이벤트, 시공간 MMDBMS 복제 요청 이벤트 등이 있다. 또한, 모니터링 관리자로부터 전달받은 정보에 따라 특정 이벤트를 발생시켜서 이벤트간의 연관관계를 관리한다. 그림 6은 이벤트간의 연관관계를 보여준다.

그림 6을 살펴보면 CPU 부하 이벤트 발생시 안정적인 시공간 미들웨어의 연동을 위해 자동적으로 필터링을 수행하여 CPU의 부하율을 낮출 수 있도록 설정해 두었다면 추가로 데이타 스트림 필터링 이벤트가 발생된다. 또한, 데이타 스트림을 서버 시공간 MMDBMS에 저장하는 요청이 들어와 데이타 스트림 저장 이벤트가 발생되고, 해당 태스크를 수행하던 중 디스크 용량이 부족한 특정 이벤트가 발생될 수 있다. 이를 해결하기 위해 필터링 후 데이타 스트림 저장 요청 이벤트가 발생되거나 데이타 스트림 저장 종료 이벤트가 발생된다.

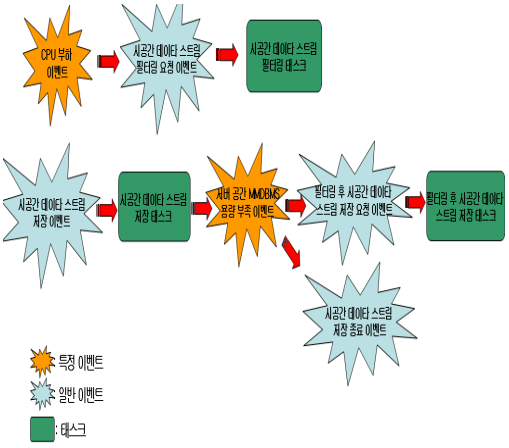


그림 6. 이벤트간의 연관관계

3.4.2 태스크 관리자

태스크 관리자는 특정 이벤트 발생 시 효율적인 태스크의 수행을 위해 해당 태스크를 생성, 삭제, 정지, 재개하는 기능을 제공한다. 또한, 태스크가 수행되는 도중 에러 발생시 이를 알려줌으로써 태스크의 수행 상태를 파악할 수 있다. 표 4는 특정 이벤트 발생시 태스크가 관리되는 내용을 보여준다.

표 4. 특정 이벤트 발생시 태스크 관리

특정 이벤트	태스크 관리
시공간 미들웨어의 CPU 부하 이벤트	데이타 스트림 필터링 태스크 생성, 정지, 삭제
시공간 MMDBMS의 디스크 용량 부족 이벤트	데이타 스트림 저장 태스크 정지, 삭제 데이타 스트림 필터링 태스크 생성, 정지, 삭제
시공간 미들웨어의 디스크 용량 부족 이벤트	복제 파일 저장 태스크 정지, 삭제
모바일 시공간 MMDBMS의 최대 접속 수 도달 이벤트	접속 요청을 받는 태스크 정지, 재개

표 4를 살펴보면 모바일 시공간 DBMS의 최대 접속 수 도달 이벤트가 발생할 경우 추가적인 접속 요청을 받지 못하므로 이를 수행하는 태스크를 중지시키고, 차후 기존에 접속된 모바일 시공간 DBMS의 접속이 종료되면 추가적인 접속이 가능하므로 다시 해당 태스크를 재개시킨다.

3.4.3 세션 관리자

세션 관리자는 각 시공간 DBMS와 시공간 미들웨어간의 안정적인 연동을 위해 서비스 요청에 따른 접속 지연과 접속된 세션의 상태를 관리하는 기능을 제공한다. 시공간 미들웨어는 각 시공간 DBMS에서의 접속 요청시 설정된 네트워크 지연 시간이 지나도 접속되지 않으면 접속 요청을 거절한다. 또한, 접속된 세션이 설정된 유효 시간동안 일련의 작업을 수행하지 않으면 세션의 연결을 종료시키고, 유효 시간이 초과되기 전에 작업이 이루어지는 경우 세션의 유효 시간을 초기화하여 관리한다. 이는 모바일 시공간 DBMS의 최대 접속 수의 제한이 있으므로 작업이 이루어지지 않는 세션과 계속 연결을 유지하는 것은 메모리를 낭비하기 때문에 이를 방지하기 위한 것이다. 네트워크 응답 시간과 세션 상태를 관리함으로써 각 시공간 DBMS와 시공간 미들웨어간의 유기적인 작업 수행이 가능하다.

3.5 자원 관리

자원 관리는 메타데이타 관리자, 모니터링 관리자, 복제 관리자로 구성된다.

3.5.1 메타데이타 관리자

메타데이타 관리자는 시공간 미들웨어에 등록된 모바일 시공간 DBMS와 서버 시공간 MMDBMS의 메타 데이타를 관리하는 기능을 제공한다. 즉, 각 시공간 DBMS의 데이타베이스 정보, 서버 시공간 MMDBMS의 복제된 데이타 정보, 서버 시공간 DBMS마다 할당된 복제 데이타의 저장 용량 정보를 관리한다.

3.5.2 모니터링 관리자

모니터링 관리자는 시공간 미들웨어와 시공간 DBMS들의 상태를 지속적으로 감시하는 기능을 제공한다. 즉, 해당 모니터링 요건이 감지되면 이를 이벤트 관리자에게 전달하여 특정 이벤트가 발생될 수 있게 한다. 모니터링 관리자가 주기적으로 감시하는 내용은 서버 시공간 MMDBMS의 디스크 용량, 시공간 미들웨어의 디스크 용량, 시공간 미들웨어의 CPU 부하율, 모바일 시공간 DBMS의 접속 수이다.

시공간 미들웨어와 서버 시공간 MMDBMS는 디스크 용량의 한계가 있기 때문에 각각 복제된 데이타나 데이타 스트림을 저장할 수 있는 충분한 디스크 용량이 있는지 확인할 필요가 있다. 또한, CPU의 과부하로 인해 최악의 경우 시공간 미들웨어가 정상적으로 동작하지 못하는 경우가 발생할 수 있기 때문에 시공간 미들웨어의 CPU 부하율을 주기적으로 확인해야 한다. 그리고 다수의 모바일 시공간 DBMS와의 작업도 시공간 미들웨어에 부담을 주기 때문에 최대 접속 수를 설정하여 제한함으로써 안정적인 시공간 미들웨어의 수행이 가능해야 한다. 표 5는 모니터링 요건에 따른 처리 방식을 보여준다.

표 5를 살펴보면 모니터링 관리자가 모니터링 감시 요건을 탐지하면 이를 이벤트 관리자에게 전달하여 처리될 수 있게 한다.

표 5. 모니터링 요건에 따른 처리 방식

모니터링 대상	감시 요건
서버 시공간 MMDBMS의 디스크 용량	디스크 용량이 일정 용량에 도달하는지 여부
시공간 미들웨어의 디스크 용량	디스크 용량이 일정 용량에 도달하는지 여부
시공간 미들웨어의 CPU 부하율	CPU 점유율이 일정 수치에 도달하는지 여부
모바일 시공간 DBMS의 접속 수	모바일 시공간 DBMS가 최대 접속 가능한 수에 도달하는지 여부

3.5.3 복제 관리자

복제 관리자는 모바일 시공간 DBMS와 서버 시공간 MMDBMS의 안정적인 복구를 지원하기 위해 복제된 데이타를 시공간 미들웨어에 저장하여 관리하는 기능을 제공한다. 서버 시공간 MMDBMS의 데이타가 시공간 미들웨어에 복제되어 저장됨으로써 차후 시스템 이상으로 인한 복구시 안정적인 지원이 가능하다.

복제된 데이타는 메타데이타 관리자를 통해 관리되고 차후 서버 시공간 MMDBMS에서 복제된 데이타 요청시 해당 복제 데이타를 전달하게 된다. 또한, 복제 데이타 저장시 무분별한 복제를 막기 위해 각 서버 시공간 MMDBMS마다 저장할 수 있는 복제 데이타의 용량을 할당하여 제한을 둔다. 이러한 정보는 복제된 데이타 저장시 메타데이타 관리자에게 전달되고, 잔여 용량이 부족하면 이를 시공간 미들웨어의 관리자에게 전달한다.

4. 시스템 구현 및 검증

본 장에서는 시스템의 구현 환경을 살펴보고, 시공간 미들웨어의 주요 기능에 대하여 상세히 설명한다. 또한, 시공간 미들웨어의 효용성을 검증하기 위하여 실시간 위치 추적 시스템에 적용한 내용에 대해서 기술한다.

4.1 시스템 구현

본 논문에서는 시공간 미들웨어를 구현하기 위하여 운영체제는 Fedora 4(Linux 2.6.15) 리눅스 환경을 기반으로 하였고, 개발 도구는 리눅스에서 제공하는 g++ 3.2.3 버전을 사용하였다. 클라이언트는 Windows XP Professional 환경을 기반으로 하였고, 개발 도구는 Java 1.5 버전을 사용하였다. 그리고 시공간 데이타 처리를 위한 API를 제공하는 GEOS는 2007년 11월에 공개된 GEOS 2.4.3 버전을 사용하였다.

시공간 함수를 처리하기 위해서는 시공간 함수가 포함된 연속 질의를 분석하고 올바른 실행 계획을 생성해야 한다. 질의문에 포함된 시공간 함수는 데이타 처리 관리자를 통해 분석된 후 실행 계획으로 변환되어 수행되며, 적절한 GEOS의 시공간 함수를 호출하여 처리한 뒤 그 결과를 반환하는 기능뿐만 아니라 기존 STREAM이 갖

고 있던 일반 산술 연산 기능도 포함하고 있다. 표 6은 시공간 질의 처리를 실험하기 위해 클라이언트에 등록한 스트림 목록을 보여준다.

표 6. 스트림 목록

이 름	타 입	속 성
Person	스트림	id Int, time Int, location Char(30)

표 6을 살펴보면 Person은 이동체의 정보를 담고 있는 스트림으로서 이동체의 고유 아이디를 나타내는 id, 시간을 나타내는 time, 그리고 위치 좌표를 나타내는 location 속성으로 구성된다. 표 7은 시공간 질의 목록을 보여준다.

표 7. 시공간 질의 목록

	예 시
시공간 질의	Select Person.id, Building.id, ST_Distance(GeomFromText(Building.location, 0), GeomFromText(Person.location, 0)) From Person, Building;

표 7을 살펴보면 시공간 질의는 동일한 시간에 이동체가 건물에서 떨어져 있는 거리를 반환하는 질의이다. 그림 7은 시공간 질의를 처리하기 위하여 스트림과 시공간 질의를 등록하는 화면을 보여주고 있다.

그림 7. 스트림과 시공간 질의 등록 화면

그림 7을 살펴보면 스트림 등록 버튼을 통해 스트림을 등록하는 다이얼로그 창을 띄워 스트림명, 속성명, 그리고 속성 정보를 입력하여 스트림을 등록하고, 질의 등록 버튼을 통해 질의를 등록하는 다이얼로그 창을 띄워 시공간 질의를 등록할 수 있다. 그림 8은 시공간 질의에 대한 질의 결과 화면을 보여준다.

그림 8을 살펴보면 시공간 질의 결과로 id가 4인 이동체와 id가 6인 건물과의 거리차를 확인할 수 있다. 또한, 시공간 미들웨어는 클라이언트를 통해 입력되는 데이타 스트림에 대한 필터링을 수행한다. 그림 9는 데이타 스트림을 필터링 하기 위해 필터링 메뉴를 띄워서 필터링을 위한 조건을 입력하는 화면을 보여준다. 사용자가 필터링을 수행할 스트림명, 속성명, 특정 범위 정보를 입력하면 해당 정보는 시공간 미들웨어에 전달된다.

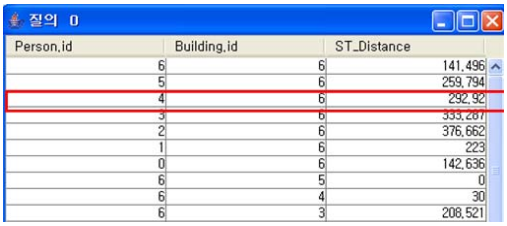


그림 8. 시공간 질의 결과 화면

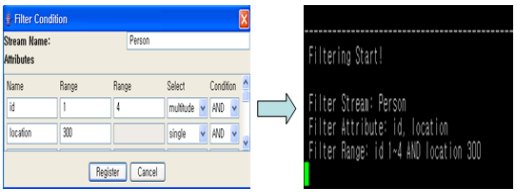
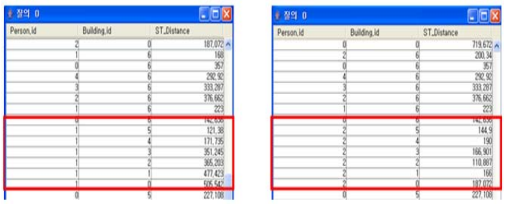


그림 9. 필터링 조건 입력 화면

그림 9를 살펴보면 Person 스트림에 id가 1부터 4까지의 범위를 가지고 위치 거리차가 300 이하인 조건들이 AND 연산으로 설정되었다. 그림 10은 시공간 질의를 수행하여 필터링하기 전의 질의 결과와 필터링이 수행되는 도중의 질의 결과 화면을 보여준다.



〈 필터링을 하기 전의 질의결과 〉 〈 필터링이 수행되는 도중의 질의 결과 〉

그림 10. 필터링 전후 질의 결과 화면

그림 10에서 보듯이 데이타 스트림 중에서 아이디가 1에서 4에 포함되고 좌표 거리차가 300 이하인 데이타 스트림은 필터링되어 버려지기 때문에 시공간 데이타 스트림의 과부하를 줄일 수 있다.

시공간 미들웨어는 모바일 시공간 DBMS와 서버 시공간 MMDBMS간의 양방향 동기화를 지원하며 이전 동기화 시점 이후부터 동기화를 수행하는 기법을 사용한다. 그림 11은 서버 시공간 MMDBMS에서 최근 데이타 우선 방식으로 모바일 시공간 DBMS에 동기화를 요청하여, 동기화 요청에 대한 정보가 모바일 시공간 DBMS에 전달된 화면을 보여준다.

그림 11과 같이 모바일 시공간 DBMS에 동기화 요청 정보가 전달되면 모바일 시공간 DBMS 사용자는 동기화

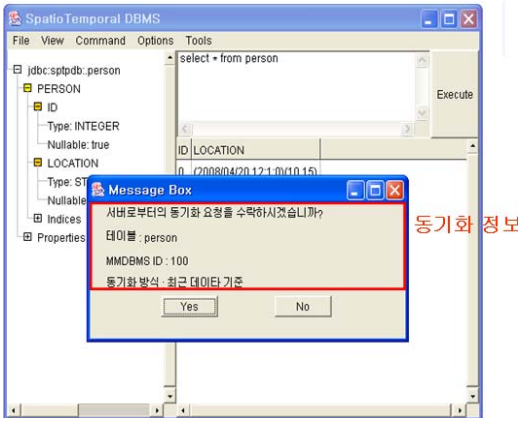


그림 11. 동기화 요청 정보 화면

를 할 것인지 판단해야 한다. 모바일 시공간 DBMS에서 동기화 요청을 수락하면 각 시공간 DBMS는 트랜잭션 정보 로그를 시공간 미들웨어에 전송하게 되고, 시공간 미들웨어는 전송된 트랜잭션 정보 로그를 비교하여 각 시공간 DBMS에 반영될 질의문을 생성하게 된다. 그림 12는 전송된 시공간 DBMS들의 트랜잭션 정보 로그를 비교하여 각 시공간 DBMS에 반영될 질의문이 생성되는 과정을 보여준다.

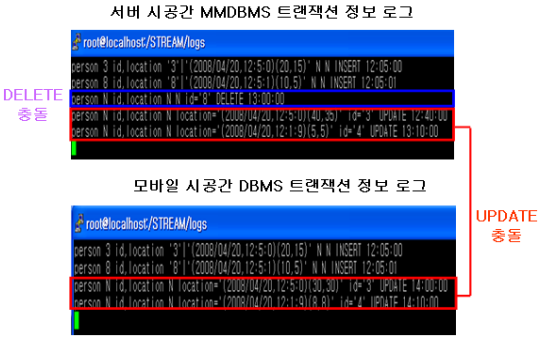


그림 12. 트랜잭션 정보 로그 비교 과정

그림 12에서 트랜잭션 정보 로그를 비교해 보면 delete 충돌과 update 충돌이 발생한 것을 확인할 수 있다. 이를 해결하기 위해 트랜잭션 수행 시간을 비교한 결과 delete 충돌은 서버 시공간 MMDBMS에서 수행된 delete 문을 모바일 시공간 DBMS에 반영하고, update 충돌은 모바일 시공간 DBMS에서 수행된 update 문을 서버 시공간 MMDBMS에 반영한다. 반영될 질의문이 정상적으로 각 시공간 DBMS에 전달되면 동기화는 완료되고, 이에 대한 정보는 동기화 완료 정보 로그에 기록된다. 그림 13은 동기화 완료 후 각 시공간 DBMS에 전송된 질의문이 반영되어 서버 시공간 MMDBMS와 모바일 시공간 DBMS간의 데이타가 일치된 모습을 보여준다.

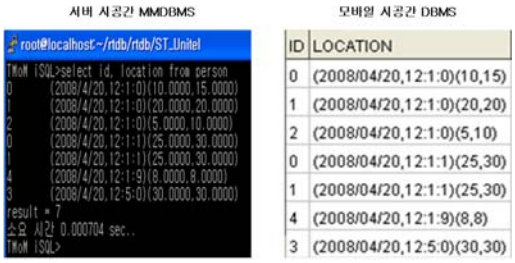


그림 13. 동기화 후 각 시공간 DBMS의 화면

4.2 시스템 검증

실시간 위치 추적 시스템인 RTLS(Real-Time Location System)는 주로 근거리 및 실내와 같은 제한된 공간에서의 위치 확인 및 위치 추적 시스템이다. 이러한 실시간 위치 추적 시스템을 통해 시공간 미들웨어의 효용성을 입증하기 위한 시나리오는 다음과 같다.

사무실 직원과 방문자의 위치 데이타는 실시간에 데이타 스트림으로 입력되고, 급작스럽게 발생할 수 있는 회사의 기밀 정보 유출에 빠르게 대응하기 위해 보안구역이 관심의 대상이 된다. 본 논문에서는 실시간으로 입력되는 사무실 직원의 위치 데이타를 모니터링하여 회사의 기밀 정보 유출에 신속하게 대응하는 시나리오를 적용해 보았다. 그림 14는 시공간 미들웨어를 실시간 위치 추적 시스템에 적용시킨 구조를 보여준다.

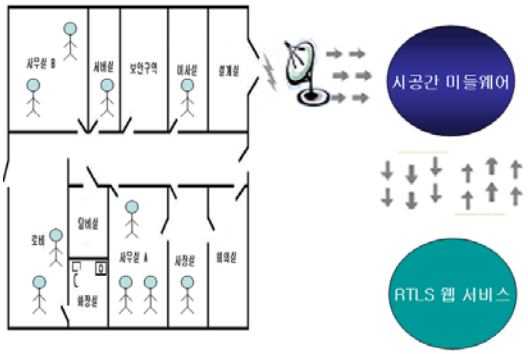


그림 14. 실시간 위치 추적 시스템

본 시나리오를 실험하기 위해 사용한 데이타는 시공간 데이타를 생성할 수 있는 데이타 생성기를 사용하였다[14]. 데이타 생성기는 센서의 특징을 고려하여 시공간 데이타를 생성하기 위해 개발된 것으로써 몇 가지 파라미터를 입력해서 일반적으로 사용되는 정규분포나 비정규분포 혹은 균등분포를 따르는 데이타를 생성한다. 그림 15는 데이타 생성기를 통하여 이동체에 대한 센서 데이타를 생성하기 위한 파라미터를 보여주는 화면이다.

그림 15를 살펴보면 파라미터는 텍스트 화일에 데이타를 저장하고, 랜덤함수의 시드(Seed)값으로 14251312를

```
DataSet.name            = stream_R.txt
DataSet.seed            = 14251312
DataSet.FirstID         = 0
DataSet.nS              = 1
DataSet.loc.x.(min, max)                = 0, 1000
DataSet.loc.y.(min, max)                = 0, 1000
DataSet.nSS             = 50
DataSet.nG              = 1
DataSet.G[0].nNodes             = 50
DataSet.G[0].v_loc.v_x.init     = r, 0, 1000
DataSet.G[0].v_loc.v_x.d        = g_100_0.2, 0, 200
DataSet.G[0].v_loc.v_x.(min,max,Adj)= 0, 1000, m
DataSet.G[0].v_loc.v_x.e        = 0, 1000
DataSet.G[0].v_loc.v_y.init     = r, 0, 1000
DataSet.G[0].v_loc.v_y.d        = g_100_0.2, 0, 200
DataSet.G[0].v_loc.v_y.(min,max,Adj)= 0, 1000, m
DataSet.G[0].v_loc.v_y.e        = 0, 1000
```

그림 15. 이동체 센서 데이타 생성 파라미터

사용하며, 센서 id는 0번부터 시작된다. 스냅샷 수는 50회이며, 50개의 센서(nNodes)가 있는 하나의 그룹(nG) 데이타를 생성한다. x와 y좌표는 0~1000 사이에서 랜덤하게 초기 위치가 정해져서 중심을 100, 편차를 0.2, 최소 및 최대를 0과 200으로 하는 정규 분포를 따라 다음 위치가 결정된다. 그림 16은 데이타 생성기가 생성한 이동체의 센서 데이타와 사무실 영역의 공간 데이타를 보여준다.

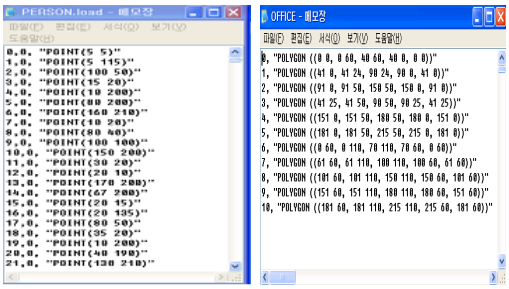


그림 16. 이동체 센서 데이타/사무실 영역 공간 데이타

그림 16을 살펴보면 이동체 50명에 대한 스냅샷 50회분량 데이타와 고정된 사무실 영역 11구역에 대한 시공간 데이타를 생성한 것을 알 수 있다.

본 논문에서는 이동 객체의 상태 및 위치 데이타를 검색하여 보여주기 위해 웹 기반으로 실시간 위치 추적 시스템을 구현하여 실험하였다. 그림 17은 실시간 위치 추적 시스템의 화면 구성을 보여준다.

사용자가 질의를 입력하기 위해 FilterBy 영역을 통해 사무실 영역의 좌하단, 우상단 좌표값을 입력하면 실시간 위치 추적 시스템은 이를 SOAP 메시지로 작성한다. 그림 18은 질의 처리를 위해 사용자가 검색하고자 하는 영역을 입력하고, 해당 내용이 SOAP 메시지로 작성되는 화면을 보여준다.

그림 18을 살펴보면 검색 영역은 '(61, 60), (150, 110)'로 입력되었고, 이를 바탕으로 SOAP 메시지가 생성된 것을 알 수 있다. SOAP 메시지는 연속 질의인 CQL로변환되어 시공간 미들웨어에 전달되며 질의가 처리된 후

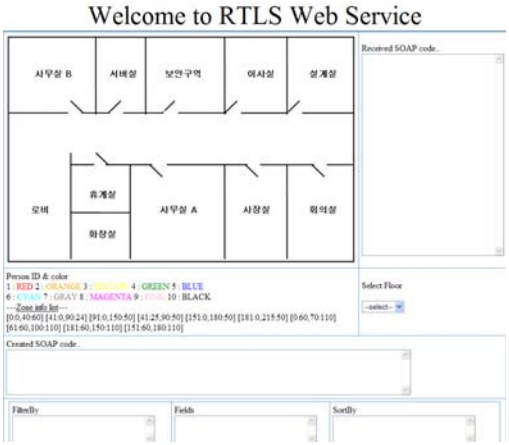


그림 17. 실시간 위치 추적 시스템 화면

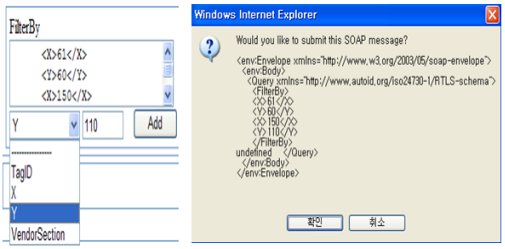


그림 18. 검색 영역 입력 및 SOAP 메시지 생성

그림 19. 실시간 위치 추적 시스템 결과 화면

반환되는 질의 결과를 Java Applet 영역에 보여주게 된다. 그림 19는 실시간 위치 추적 시스템의 결과 화면을 보여준다.

그림 19를 살펴보면 입력된 좌표에 따른 검색영역이 '서버실'과 '보안구역'이고, 현재 해당 영역에 위치한 이동체들이 화면에 표시되는 것을 보여준다.

5. 결 론

최근 유비쿼터스 컴퓨팅 환경에서 다수의 컴퓨팅 디바이스가 생성하는 대량의 데이타 스트림을 가공하여 사용자 또는 응용 서비스에게 전달해주는 미들웨어 기술이 요구되고 있다. 하지만, 기존의 데이타 스트림 처리 기술은 시공간 데이타에 대한 질의 처리를 효율적으로 지원하지 못하기 때문에 시공간 데이타를 다루는 미들웨어의 연구 개발 실정이 매우 미흡한 상태이다.

따라서, 본 논문에서는 이를 해결하기 위해 시공간 데이타 스트림의 효율적인 처리 및 관리를 지원하고 모바일 시공간 DBMS와 서버 시공간 MMDBMS간의 상호운용성을 제공하기 위한 시공간 미들웨어를 설계 및 구현하였다. 특히 시공간 미들웨어는 스탠포드 대학의 STREAM을 기반으로 시공간 질의 처리가 가능하도록 확장하였다. 시공간 미들웨어는 입력 부하를 방지하기 위한 스트림 관리 기술, 모바일 시공간 DBMS와 시공간 MMDBMS간의 양방향 동기화를 통해 질의 처리시 데이타의 무결성을 보장하기 위한 동기화 관리 기술, 각 시공간 DBMS와의 접속된 세션의 상태를 관리하기 위한 실시간 관리 기술, 시공간 미들웨어와 각 시공간 DBMS의 상태를 지속적으로 감시하여 안정적인 연동이 가능하기 위한 자원 관리 기술을 지원한다.

마지막으로, 본 논문에서 개발한 시공간 미들웨어를 실시간 위치 추적 시스템에 적용해 본 결과 지속적으로 입력되는 데이타 스트림에 대한 시공간 연산을 실시간으로 처리할 수 있음을 보여주었다. 이를 통해 본 시스템이 유비쿼터스 환경에서 실시간 관리가 필요한 다양한 응용분야에서 유용하게 사용될 수 있음을 확인하였다.

참 고 문 헌

[1] 노현숙, 고병열, 박현우, “센서: u-센서 네트워크 산업의 활성화 전략,” 한국과학기술정보연구원, 2004.

[2] 대전광역시첨단산업진흥재단, 유비쿼터스 환경의 센서 데이타 스트림 처리 기술, http://www.ntb.or.kr/front/techTrans/SaleTechView.asp?techCode=S2005003259&seq=158.

[3] 김민수, 이용준, 박종현, “USN 미들웨어 기술개발 동향,” 전자통신동향분석, 제22권 제3호, 2007, pp.67-79.

[4] 윤희용, “유비쿼터스 컴퓨팅 미들웨어 기술,” 전자공학회지, 제30권 제11호, 2003, pp. 102-111.

[5] 정혜선, 정창성, “u-city를 위한 유비쿼터스 컴퓨팅 미들웨어,” 인터넷정보학회지, 제7권 제2호, 2006, pp. 38-45.

[6] 강홍구, 박치민, 홍동숙, 한기준, “공간 센서 데이타의 효율적인 실시간 처리를 위한 공간 DSMS의 개발,” 한국공간정보시스템학회 논문지, 제9권 제1호, 2007, pp. 45-57.

[7] 신중수, 김정준, 강홍구, 한기준, “대용량 지도 서비스를 위한 공간 미들웨어의 설계 및 구현,” 2006년 추계학술대회 논문집, 한국공간정보시스템학회, 2006, pp .275-280.

[8] 원종호, 이미영, 김명준, “유비쿼터스 컴퓨팅 환경을 위한 RFID 기반 센서 데이타 처리 미들웨어 기술 동향,” 전자통신동향분석, 제19권 제5호, 2004, pp. 21-29.

[9] 이기영, 김동오, 신중수, 한기준, “대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어의 개발,” 한국공간정보시스템학회 논문지, 제10권 제1호, 2008, pp. 1-14.

[10] Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J., STREAM: The Stanford Data Stream Management System, http://dbpubs.stanford.edu /pub/ 2004-20, 2004.

[11] Open Geospatial Consortium Inc., OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option, 2005.

[12] Arasu, A., Babu, S., and Widom, J., “The CQL Continuous Query Language: Semantic Foundations and Query Execution,” The VLDB Journal, Vol.15, No.2, 2006, pp. 121-142.

[13] Ramsey, P., The State of Open Source GIS, Refractions Research Inc., 2006.

[14] 박치민, 김동오, 홍동숙, 한기준, “이동 센서를 위한 데이타셋 생성기,” GIS/RS 공동 춘계학술대회논문집, 한국GIS학회, 2006, pp. 131-137.

김 정 준

2003년 건국대학교 컴퓨터공학과(공학사)
2005년 건국대학교 대학원 컴퓨터공학과(공학석사)
2005년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야는 공간 데이타베이스, GIS, LBS, 텔레매틱스, USN

정 연 종

2007년 건국대학교 컴퓨터공학과(공학사)
2009년 건국대학교 대학원 컴퓨터공학과(공학석사)
관심분야는 공간 메인 메모리 데이타베이스, GIS, 미들웨어

김 동 오

2000년 건국대학교 컴퓨터공학과(공학사)
2002년 건국대학교 대학원 컴퓨터공학과 (공학석사)
2006년 건국대학교 대학원 컴퓨터공학과 (공학박사)
2006년~현재 건국대학교 컴퓨터공학부 강의교수
관심분야는 데이타베이스, LBS, GML, 유비쿼터스 센서 네트워크, 정보시스템 감리

한 기 준

1979년 서울대학교 수학교육학과(이학사)
1981년 한국과학기술원(KAIST) 전산학과 (공학석사)
1985년 한국과학기술원(KAIST) 전산학과 (공학박사)
1985년~현재 건국대학교 컴퓨터공학부 교수
1990년 Stanford 대학 전산학과 Visiting Scholar
2000년~2002년 한국정보과학회 데이타베이스연구회 운영위원장
2004년~2006년 한국공간정보시스템학회 회장
2004년~2008년 한국정보시스템감리사협회 회장
관심분야는 공간 데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리