공간 센서 데이타의 효율적인 실시간 처리를 위한 공간 DSMS의 개발†

Development of a Spatial DSMS for Efficient Real-Time Processing of Spatial Sensor Data

강홍구* / Hong-Koo Kang 박치민** / Chi-Min Park
홍동숙*** / Dong-Suk Hong 한기준**** / Ki-Joon Han

요 약

최근 센서 장치의 발달은 무선 센서 네트워크와 같은 진보된 기술에 대한 연구를 가속화시켰다. 뿐만 아니라, GPS(Global Positioning System) 기능을 보유한 공간 센서는 공간 정보와 비공간 정보를 동시에 사용하는 유비쿼터스 시대를 이끌어가고 있다. 이러한 새로운 시대에 있어서, 공간 센서 데이타에 대한 실시간 처리 시스템은 필수적이다. 이 때문에, DSMS(Data Stream Management System)라 불리는 새로운 개념의 데이타 처리 시스템이 많은 연구자들에 의해 연구되고 있다. 그러나 대부분의 DSMS들이 공간 센서 데이타 처리를 위한 공간 함수를 지원하지 않기 때문에 유비쿼터스 컴퓨팅에 적합하지 않다. 그러므로, 본 논문에서는 이러한 문제를 해결하고자 STREAM(STanford stREam datA Manager)을 확장한 공간 DSMS를 설계 및 구현하였다. 즉, Geometry 타입과 공간 함수를 공간 센서 데이타 처리를 위해 추가하였고, 또한 시스템에서 공유되는 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리자를 추가하였다. 특히, 상호운용성을 위하여 OGC(Open Geospatial Consortium)에서 제시한 Simple Features Specification for SQL을 구현하였고, GEOS(Geometry Engine Open Source)의 알고리즘을 본 시스템에 적용하였다.

Abstract

Recently, the development of sensor devices has accelerated researches on advanced technologies like Wireless Sensor Networks. Moreover, spatial sensors using GPS lead to the era of the Ubiquitous Computing Environment which generally uses spatial information and non-spatial information together. In this new era, a real-time processing system for spatial sensor data is essential. In this reason, new data processing systems called DSMS(Data Stream Management System) are being developed by many researchers. However, since most of them do not support geometry types and spatial functions to process spatial sensor data, they are not suitable for the Ubiquitous Computing Environment. For these reasons, in

† 본 연구는 서울시 산학연 협력사업의 신기술 연구개발 지원사업의 지원으로 수행되었음.
■ 논문접수 : 2007.2.9 ■ 심사완료 : 2007.4.25
* 교신저자 건국대학교 컴퓨터공학부 (hkkang@db.konkuk.ac.kr)
** 건국대학교 컴퓨터공학부 (cmpark@db.konkuk.ac.kr)
*** 건국대학교 컴퓨터공학부 (dshong@db.konkuk.ac.kr)
**** 건국대학교 컴퓨터공학부 (kjhan@db.konkuk.ac.kr)

this paper, we designed and implemented a spatial DSMS by extending STREAM which stands for STanford stREam datA Manager, to solve these problems. We added geometry types and spatial functions to STREAM in order to process spatial sensor data efficiently. In addition, we implemented a Spatial Object Manager to manage shared spatial objects within the system. Especially, we implemented the Simple Features Specification for SQL of OGC for interoperability and applied algorithms in GEOS to our system.

주요어 : 데이타 스트림, 공간 DSMS, 공간 센서 데이타, 실시간 처리
Keyword : Data Stream, Spatial DSMS, Spatial Sensor Data, Real-Time Processing

1. 서 론

최근 유비쿼터스 컴퓨팅 환경에 대한 관심과 연구가 집중되면서 다양한 센서들이 연구되고 있다. 특히 RFID나 GPS 등을 통한 위치 측정 기술이 발달하면서 기존에 센서를 통하여 측정 대상물에 대한 온도, 습도, 음향과 같은 상태 정보만을 다루던 단계를 넘어서 해당 값이 측정된 위치를 함께 다루는, 즉 공간 데이타와 비공간 데이타를 같이 다루는 공간 센서 데이타에 대한 연구들이 활발히 진행되고 있다[1,2,3]. 특히 수집된 공간 센서 데이타들을 실시간으로 처리하는 응용 시스템들에 대한 연구들도 활발하게 진행되고 있는데, 그 예로서 실시간 환경 감시, 침입자 탐지, 산불 감시, 재난 복구 시스템 등이 있다. 이러한 공간 센서 데이타 처리 시스템들은 실시간으로 입력되는 공간 센서 데이타에 대해 주어진 질의를 처리하여 그 결과를 실시간으로 제공하여야 한다[4].

센서들로부터 수집되는 공간 센서 데이타는 매우 빠른 속도로 추가되고 한번 추가된 데이타는 삭제되거나 수정되지 않는 특징이 있다. 이러한 특징을 갖는 데이타를 실시간으로 처리하기 위해 데이타 스트림(Data Stream)이라는 새로운 형태의 데이타 모델에 대해 연속 질의(Continuous Query)를 처리할 수 있는 DSMS(Data Stream Management System)가 필요하게 되었다[5,6]. 이러한 DSMS는 디스크나 메모리에 저장되어 있는 모든 데이타를 질의 처리 대상으로 하는 기존 DBMS와는 다르게, 최근 데이타만을 처리하거나 과거부터 현재까지 입력된 데이타에 대한 요약 정보만을 처리한다. 예로 MAX()와 같은 집계 함수를 계산할 때 기존 DBMS에서는 과거에 입력됐던 데이타가 변경될 경우 최대값을 다시 계산하기 위해 과거 데이타를 다시 검색해야 한다. 그러나 DSMS에서는 과거에 입력된 값은 추가 및 변경되지 않기 때문에 최대값을 계산하기 위해서는 새로 입력된 데이타만을 고려하면 된다.

최근 몇 년간 데이타 스트림을 처리하기 위하여 여러 대학과 연구 기관에서 DSMS에 대한 연구가 활발하게 진행되고 있다. 이러한 연구로는 인터넷에서 발생하는 이벤트에 대한 모니터링을 효율적으로 하기 위해 연산자 공유를 지원하는 NiagaraCQ[7], 연속 질의에 필요한 연산자들을 Box 개념으로 나눠서 질의를 구성하는 Aurora/Borealis[8], 하나의 스트림을 대상으로 하는 질의를 데이타 스트림의 변화에 따라 적응적으로 처리하는 기법을 연구한 Gigascope[9], 조인, 집합 함수 등의 상태 정보 공유를 통해 처리 효율을 높인 TelegraphCQ[10], 데이타 스트림 처리를 DBMS의 관점에서 접근하여 스트림과 릴레이션을 정의한 STREAM[11], 그리고 센서 네트워크 환경을 위한 DSMS로서 많은 수의 센서 간에 이루어지는 Join을 효율적으로 처리하기 위한 Nile[12] 등이 있다.

그러나 이러한 연구들은 주로 비공간 속성에 대한 질의의 효율 향상이나 적응적 처리에 중점을 두며 공간 속성에 대한 처리를 고려하고 있지 않는 문제점이 있다. 공간 속성과 비공간 속성을 모두 갖

고 있는 센서 데이타 스트림의 경우에는 공간 속성에 대한 연산이 전체 시스템의 효율을 높이는데 중요한 역할을 한다. 예를 들어, 기름띠로 오염된 바다의 수온 변화를 연구할 경우 바다의 수온을 측정하기 위하여 설치된 센서 중에서 기름띠의 범위 안에 있는 센서들로부터 수집된 데이타만이 관심 대상이 될 것이다. 기존 연구들은 공간 속성을 고려하지 않아서 전체 센서 중에서 필요한 센서 값만을 효율적으로 처리하기 힘든 문제가 발생하였다.

본 논문에서는 공간 센서 데이타를 포함하는 공간 데이타를 실시간으로 처리하기 위하여, 스탠포드 대학에서 개발한 STREAM을 기반으로 공간 데이타를 처리하기 위해 공간 객체 타입과 공간 함수, 공간 객체 관리자 등을 추가한 공간 DSMS를 설계 및 구현하였다. 특히, 시스템의 상호운용성을 고려하여 Open Geospatial Consortium(OGC)에서 제시한 Simple Feature Specification[13]에 따라 공간 DSMS에 공간 객체 타입 및 함수를 추가하였으며, 공간 함수의 구현에서는 Geometry Engine Open Source (GEOS)[14]를 사용하였다. 또한, 여러 가지 형태의 공간 데이타를 처리하기 위해 STREAM에 Geometry(공간 객체) 타입을 추가하였고, 여러 개의 질의가 동시에 처리될 때 여러 관계 연산자에 의해 공유되는 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리자를 개발하였다.

본 논문의 구성은 다음과 같다. 2장에서 본 논문에서 다루는 주요 개념인 DSMS에 대해서 간단히 소개하고, 스탠포드 대학에서 개발한 STREAM에 대하여 살펴본다. 3장에서는 공간 DSMS의 전체 구조, 공간 데이타 타입, 공간 함수 처리기, 공간 객체 관리자의 개발에 대하여 설명하고, 4장에서는 가상 시나리오 실행 결과를 통해 본 시스템의 유용성을 검증한다. 마지막으로 5장에서는 결론 및 향후 연구 과제에 대하여 기술한다.

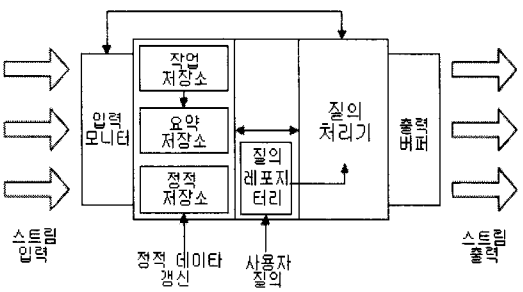
2. 관련 연구

본 장에서는 DSMS에 대한 일반적인 특징과 구조를 설명하고, 본 연구에서 공간 DSMS을 구현하기 위해 사용한 STREAM에 대하여 살펴본다.

2.1 DSMS

DSMS는 지속적으로 입력되는 데이타 스트림을 처리하기 위한 시스템이다. 이를 위해 기존에 개발된 DBMS가 갖는 가정 중 하나인 "데이타를 로드한 후에 인덱스를 생성하고 여기에 질의를 요청한다."라는 가정이 제거되었다[15]. 즉, 디스크 등의 저장소에 쌓여있는 데이타를 메모리에 로드하는 대신 실시간으로 입력되는 데이타를 처리 대상으로 하며, 질의는 한 번만 수행되는 것이 아니라 시스템에 등록된 후 연속적으로 수행된다. 이러한 DSMS는 센서 네트워크뿐만 아니라 네트워크 모니터링, 결제 정보 분석, 판매 내역 트랜잭션 분석 등과 같은 다양한 분야에서 활용이 가능하다.

일반적인 DSMS의 구조는 그림 1과 같다[6]. 그림 1에서 보듯이 일반적으로 DSMS는 지속적인 입력을 제어하기 위한 입력 모니터(Input Monitor), 연속 질의를 저장해두는 질의 레포지터리(Query Repository), 이를 지속적으로 처리하기 위한 질의 처리기(Query Processor), 시스템에 필요한 정적인 정보를 저장하기 위한 정적 저장소(Static Storage), 연속 질의 처리에 사용되는 메모리를 효율적으로 관리하기 위한 작업 저장소(Working Storage), 요약 저장소(Summary Storage), 처리 결과를 출력하기 위한 출력 버퍼(Output Buffer) 등으로 구성된다.

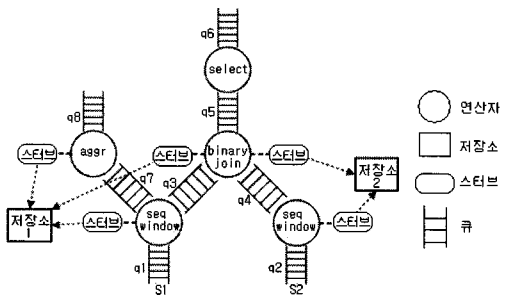


<그림 1> 일반적인 DSMS의 구조

2.2 STREAM

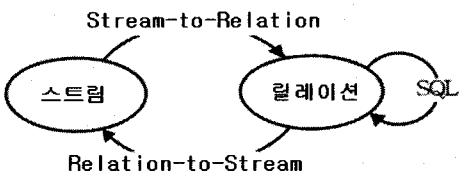
STREAM은 데이타 스트림을 모니터링하고 필터링하기 위하여 스탠포드 대학에서 개발한 DSMS이며 현재 서버 0.6 버전과 클라이언트 0.3 버전이 소스와 함께 BSD License 형태로 공개되어 있다[15]. 서버는 Linux 기반에서 C++로 구현되어 있으며 라이브러리로 사용이 가능하고, 데이타 스트림 처리를 위한 연속 질의 언어 CQL(Continuous Query Language)을 지원한다. 클라이언트는 운영체제에 독립적인 Java로 구현되어 있으며 데이타 입력 및 질의 입력, 실행 계획 뷰어 등을 제공한다.

그림 2는 STREAM의 동작 방식을 보여주고 있다[11]. 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, 연산자들은 큐로 연결되어 있어서 각 연산자는 입력 큐의 데이타들을 처리하여 출력 큐에 저장한다. 이 외에도, 하나의 질의 결과가 다른 질의들에 의해 공유되는 기능이 있어서 질의 처리 속도 및 공간 효율성을 향상시킨다.



<그림 2> STREAM의 동작 방식

STREAM은 CQL이라는 자체적으로 정의한 연속 질의 언어를 사용한다. 그림 3에서 보는 바와 같이 CQL은 스트림을 릴레이션으로 바꿔주는 Stream-to-Relation 연산자, 릴레이션을 다시 스트림으로 바꿔주는 Relation-to-Stream 연산자, 기존 관계형 DBMS에서 SQL로 처리하던 관계대수의 연산자로 구성된다[16]. 여기서 스트림이란 갱신 및 삭제는 발생하지 않고 지속적인 입력만 발생하는 데이타 모델이며, 릴레이션이란 입력과 삭제가 발생하며 유한한 크기를 갖는 데이타 모델이다. 스트림은 크기가 무한하기 때문에 기존의 유한 집합을 대상으로 하는 관계형 연산들을 수행할 수 없다. 그러므로 스트림에 슬라이딩 윈도우(Sliding Window) 기법을 적용함으로써 유한한 크기를 갖는 릴레이션으로 바꿔준 뒤 기존의 관계형 연산자들을 적용하게 된다.



<그림 3> CQL의 연산자 종류

그러나 STREAM은 공간 데이타에 대한 처리를 전혀 고려하지 않고 있으며, 이 때문에 위치나 넓이와 같은 공간 정보를 기반으로 한 연산을 수행하는데 부적합한 상태에 있다.

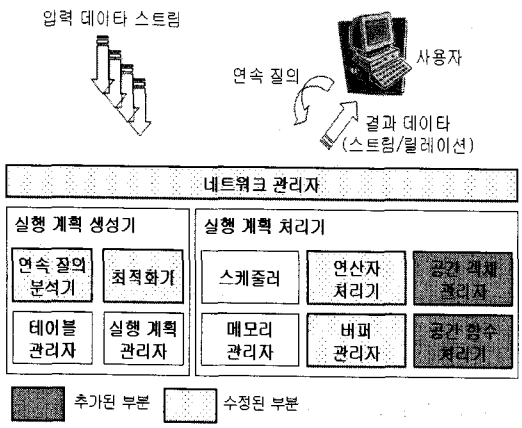
3. 공간 DSMS 개발

본 장에서는 공간 DSMS의 전체 구조, 공간 DSMS가 지원하는 공간 데이타 타입, 공간 데이타 타입에 대해서 공간 연산자를 수행하는 공간 함수 처리기, 그리고 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리자에 대하여 설명한다.

3.1 시스템 구조

공간 DSMS는 네트워크 관리자, 연속 질의 분석기, 최적화기, 테이블 관리자, 실행 계획 관리자, 스케줄러, 메모리 관리자, 연산자 처리기, 버퍼 관리

자, 공간 객체 관리자, 공간 함수 처리기로 구성된다. 그림 4는 공간 DSMS의 구조를 보여준다.



<그림 4> 공간 DSMS의 구조

네트워크 관리자는 입력 데이타 스트림으로부터 데이타를 입력받고, 사용자와 연결되어 연속 질의 요청을 접수하고 결과를 반환한다. 네트워크 관리자가 연속 질의 요청을 받으면, 실행 계획 생성기 서브시스템이 질의를 전달받아서 실행 계획으로 변환한다. 실행 계획 처리기 서브시스템은 지속적으로 입력되는 데이타에 대하여 실행 계획을 처리하고 그 결과를 네트워크 관리자를 통하여 사용자에게 실시간으로 돌려준다.

실행 계획 생성기 서브시스템의 연속 질의 분석기는 테이블 관리자에 등록된 정보를 바탕으로 질의문을 파싱하여 연산자들의 리스트를 만들고, 최적화기는 등록된 여러 질의들에 대한 연산자 공유 및 최적화를 고려하여 실행 계획을 생성한 후 실행 계획 관리자에 등록한다.

실행 계획 처리기 서브시스템의 스케줄러는 실행 계획 관리자에 등록된 여러 실행 계획들을 연산자 단위로 스케줄링하고 순차적으로 적절한 연산자 처리기를 실행시킨다. 메모리 관리자는 버퍼 관리자가 사용할 메모리를 페이지 단위로 할당 및 해제하고, 버퍼 관리자는 연산자 간 데이타 전달을 위한 큐, 처리 결과 임시 저장을 위한 저장소, 연산자의 중간 값을 저장하기 위한 스터브로 구성된다. 연산자 처리기는 큐로부터 처리해야 할 데이타를 받아와서 필요한 연산을 수행한 후 결과를 큐에 저장한다. 연산자 처리기는 공간 함수 처리 요청시 공간 함수 처리기를 호출하고, 공간 함수 처리기는 GEOS 라이브러리를 사용하여 결과를 계산한다.

본 논문에서는 기존의 연속 질의 분석기가 공간 연속 질의를 분석할 수 있도록 공간 데이타와 공간 함수 키워드를 처리하는 기능을 추가하였으며, 기존의 최적화기가 공간 연속 질의에 대한 실행 계획을 생성할 수 있도록 공간 함수의 입출력 데이타 타입과 크기 정보를 추가하였다. 또한 기존의 연산자 처리기를 수정하여 실행시간에 적절한 공간 함수를 호출할 수 있도록 하였으며, 기존의 버퍼 관리자를 수정하여 공간 객체 관리자에 등록된 공간 객체들을 고려하여 메모리를 관리할 수 있도록 하였다. 그리고 공간 DSMS가 지원하는 공간 데이타 타입과 공간 데이타 타입에 대해 공간 연산자를 수행하는 공간 함수 처리기와 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리자를 구현하여 추가하였다.

3.2 공간 데이타 타입

공간 DSMS는 공간 데이타가 갖는 다음과 같은 특징을 고려한 Geometry 타입을 제공하기 때문에 효율적으로 공간 데이타를 처리할 수 있다. 공간 데이타는 점, 선, 면 등의 다양한 형태를 갖고 하나 또는 그 이상의 점들로 구성되며 경우에 따라 여러 형태의 공간 데이타들이 모인 집합이 될 수도 있다. 이 때문에 Int나 Float와 같은 고정 크기의 데이타 타입과는 달리 저장 공간 크기의 변화가 심하다. 또한 공간 DSMS는 현재 사용할 데이타만을 저장하고 있기 때문에 기존 DBMS에서와 같이 저장에 적합한 포맷으로 저장한 후에 처리 시에 변환을 하는 방식을 사용할 필요가 있다.

Geometry 타입은 OGC에서 제시하는 12가지 타입의 공간 데이타들을 저장하는데 사용되는데, 이 중 OGC의 정의에 따라 입력이 가능한 7가지 타입의 공간 데이타들은 표 1과 같다.

<표 1> 입력이 가능한 공간 데이타의 형태 목록

공간 데이타 타입	설 명
Point	하나의 점
LineString	2개의 점으로 만들어지는 선분이 여러 개 이어져 있는 형태
Polygon	하나의 외부 테두리와 0 또는 그 이상의 내부 테두리로 면을 이루는 형태
MultiPoint	여러 개의 Point로 구성된 형태
MultiLineString	여러 개의 LineString으로 구성된 형태
MultiPolygon	여러 개의 Polygon으로 구성된 형태
GeomCollection	하나 이상의 Geometry로 구성된 형태

Geometry 타입은 GEOS가 생성하는 공간 객체에 대한 포인터이며 내부적으로 Int 값과 같은 고정 크기 컬럼으로 저장되고, 이 값을 사용할 때 Heap 영역에 저장된 공간 객체에 대한 포인터 형으로 캐스팅한다. 그러므로 저장소에 포인터에 대한 정보를 기록해 두어야 하기 때문에 저장소는 자신이 저장하는 튜플에 몇 개의 Geometry 타입 컬럼이 있는지에 대한 정보인 numGeoms와 그 컬럼들의 위치 정보인 geomCols를 갖고 있다. 예를 들어 저장소에 <아이디 Int, 중심 위치 Geometry, 넓이 Geometry, 온도 Int>로 정의된 튜플을 저장한다면, numGeoms는 2이며 geomCols[] = {1, 2}가 된다.

3.3 공간 함수 처리기

공간 센서 데이타에 있는 공간 데이타를 효율적으로 처리하기 위해서는 복잡한 연산 과정이 필요하나 기존 STREAM은 다차원 형태인 공간 데이타를 지원하지 않는다. 따라서 본 논문에서는 GEOS

<표 2> 공간 DSMS가 지원하는 공간 함수 목록

함수 종류	함수 이름	설 명
공간 객체 변환 함수	GeomFromText	WKT 형태의 문자열을 공간 객체로 바꿔줌
	AsText	공간 객체를 WKT 형태의 문자열로 바꿔줌
공간 관계 함수	Equals	두 공간 객체가 동일하면 참을 반환함
	Disjoint	두 공간 객체가 교집합이 없으면 참을 반환함
	Touches	두 공간 객체의 경계선만 닿았으면 참을 반환함
	Within	첫 번째 공간 객체가 두 번째 공간 객체를 포함하면 참을 반환함
	Overlaps	두 공간 객체가 같은 차원이면서 겹쳐 있으면 참을 반환함
	Crosses	두 공간 객체가 교차하면서 교차된 영역의 차원이 같거나 줄어들면 참을 반환함
	Intersects	두 공간 객체가 교차하면 참을 반환함
	Contains	두 번째 공간 객체가 첫 번째 공간 객체를 포함하면 참을 반환함
	Relate	두 공간 객체의 관계를 DE-9IM 문자열로 반환함
공간 분석 함수	Distance	두 공간 객체간의 최단거리를 계산함
	Intersection	두 공간 객체가 겹치는 부분을 공간 객체로 반환함
	Difference	첫 번째 공간 객체에서 두 번째 공간 객체를 뺀 부분을 반환함
	Union	두 공간 객체를 합친 공간 객체를 반환함
	SymDifference	두 공간 객체를 합친 것에서 겹치는 부분을 뺀 공간 객체를 반환함
	Buffer	주어진 공간 객체를 주어진 크기만큼 증가시킨 공간 객체를 반환함
	ConvexHull	주어진 공간 객체를 포함하면서 가장 작은 볼록 폴리곤을 반환함

[14]에서 제공하는 알고리즘을 사용하여 공간 함수를 처리하는 공간 함수 처리기를 구현하였다.

표 2는 OGC에서 정의한 것 중 공간 DSMS가 지원하는 18개의 공간 함수 종류, 함수 이름, 설명 등을 보여준다. 공간 객체 변환 함수는 WKT 형태로 제공된 공간 데이타를 공간 객체로 바꾸거나 그 반대의 작업을 하기 위한 함수들이다. 공간 데이타는 교환의 편의 및 상호운용성을 위해서 WKT와 같은 문자열을 사용하며 시스템에 저장될 때 공간 객체로 변환된다. GeomFromText는 WKT 형태의 문자열을 공간 객체로 변환하고, AsText는 공간 객체를 WKT 형태의 문자열로 변환한다. 공간 관계 함수는 두 공간 객체의 공간상의 관계를 파악하기 위한 함수들이며, 공간 분석 함수는 하나 혹은 두 공간 객체로부터 새로운 공간 객체 혹은 의미있는 값을 얻기 위한 함수들이다.

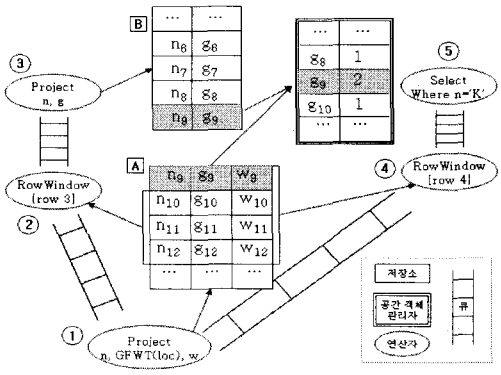
공간 함수를 처리하기 위해서는 공간 함수가 포함된 연속 질의를 분석하고 실행 계획을 생성해야 한다. 이를 위해 연속 질의 분석기는 질의문에 있는 공간 함수를 분석하고, 실행 계획 생성기는 공간 함수가 입력받거나 출력하는 Geometry 타입을 처리한다.

3.4 공간 객체 관리자

기존 STREAM은 메모리에 대한 포인터를 지원하지 않기 때문에 저장소가 더 이상 사용하지 않는 튜플을 메모리에서 해제할 때 참조 값이 적혀있는 메모리 영역만을 삭제해 주었다. 그러나 공간 객체는 Heap 영역에 생성되고 저장소에는 포인터 값만 저장되기 때문에 저장소에 저장되어 있는 포인터 정보가 메모리에서 삭제된다고 하더라도 Heap 영역에 있는 공간 객체가 메모리에서 삭제되지 않는다. 이 때문에 공간 객체를 별도로 추가/삭제하는 과정이 필요하다. 그리고 Heap 영역에 있는 공간 객체를 저장소가 직접 삭제하는 방식은 공간 객체가 여러 저장소에 동시에 존재할 경우 같은 작업을 중복 처리해야 하기 때문에 비효율적이다.

따라서 이러한 문제를 해결하기 위해 여러 저장소에서 참조하는 공간 객체를 효율적으로 관리하는 공간 객체 관리자를 개발하였다. 공간 객체 관리자는 여러 저장소에서 참조하는 공간 객체의 공간 객체 아이디와 참조 카운터를 관리함으로써 공간 객체가 사용하는 메모리 공간의 낭비를 최소화하고 중복되는 메모리 할당 및 해제 작업을 줄인다.

그림 5는 공간 객체가 여러 저장소에서 공유되는 예를 보여준다.



<그림 5> 공간 객체 공유 예

그림 5를 통하여 공간 객체 관리자의 기본적인 동작 방식을 살펴보면 다음과 같다. 연산자 1이 GFWT 함수를 통해 위치(loc)로부터 공간 객체를 생성한 뒤 이것을 이름(n)과 냉각수 잔량(w) 등과 함께 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 공간 객체 g_9가 존재하게 되며 공간 객체 관리자는 g_9에 대한 참조 카운터를 2로 늘린다. 연산자 4가 9번째 튜플을 처리하면 저장소 A에서 9번째 튜플이 삭제되는데, 이때 공간 객체 관리자는 Heap 영역에 있는 공간 객체를 바로 삭제하지 않고 참조 카운터만 1로 줄인 후, 저장소 B에서도 9번째 튜플이 삭제되면 참조 카운터가 0이 되면서 g_9를 삭제한다.

표 3은 공간 객체 관리자에 공간 객체가 추가 및 삭제되는 조건들을 보여준다. 저장소들은 저장 공간을 페이지 단위로 할당받으며, 하나의 페이지에는 여러 튜플들이 들어가게 된다. 저장소의 특성에 따라 튜플을 삭제하는 시기가 다른데 기본 저장소인 단순 저장소(Simple Store)와 Window 연산의 결과를 저장하는 윈도우 저장소(Window Store)는 페이지 단위로 삭제가 되며, 릴레이션을 저장하기 위한 릴레이션 저장소(Relation Store), 릴레이션 저장소에 인덱스가 추가된 리니지 저장소(Lienage Store), 특정 컬럼을 기준으로 분할된 윈도우를 지원하는 분할 윈도우 저장소(Partition Window Store)는 튜플 단위로 삭제된다.

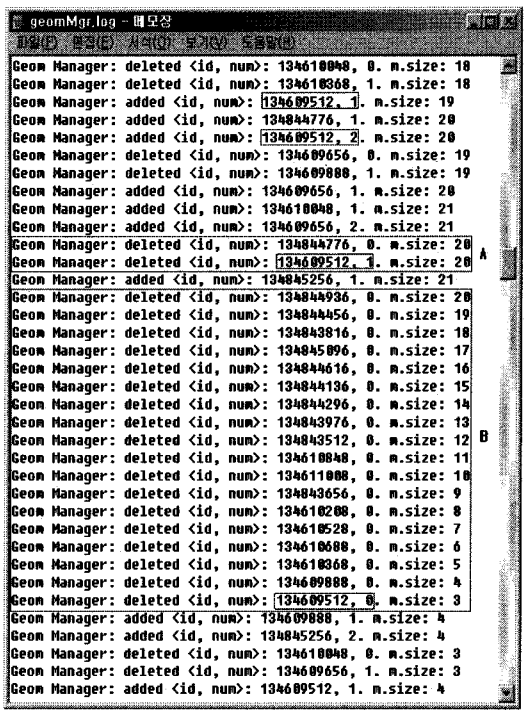
<표 3> 공간 객체 관리자의 추가/삭제 조건

조 건	추가 / 삭제
공간 객체 복사	추 가
GeomFromText 처리	
Intersection 처리	
Difference 처리	
Union 처리	
SymDifference 처리	
Buffer 처리	
ConvexHull 처리	
Window Store 페이지 삭제	삭 제
Simple Store 페이지 삭제	
Relation Store 튜플 삭제	
Partition Window Store 튜플 삭제	
Lineage Store 튜플 삭제	
질의 처리 종료	전부 삭제

공간 객체 관리자에 공간 객체를 추가하는 작업은 공간 함수가 연산 결과로 공간 객체를 생성할 때뿐만 아니라 Project 연산자를 통해 공간 객체를 복사할 때도 수행된다. 질의 처리가 종료되면 현재 생성된 공간 객체를 더 이상 사용하지 않으므로 모든 공간 객체들은 삭제된다.

그림 6은 공간 객체 관리자가 동작하는 과정을 로그로 남긴 모습이다. 각 줄에는 삭제(deleted)인지 추가(added)인지 여부, 삭제된 객체의 아이디(id), 현재 참조 카운터 개수(num)가 있고, 공간 객체 관리자에 등록된 객체 수(m.size)가 기록된다. A는 릴레이션 저장소에서 튜플 단위로 삭제되는 부분이고 B는 윈도우 저장소에서 페이지 단위로 삭제되는 부분이다. A와 같은 튜플 단위 삭제는 적은 개수에 대하여 자주 발생하며, B와 같은 페이지 단위 삭제는 많은 개수에 대하여 가끔씩 발생한다.

그림 6과 같이 134609512번 아이디를 갖는 공간 객체가 릴레이션 저장소와 윈도우 저장소에 각각 추가되면서 카운터가 2로 증가하였다. 최초에 134609512번 공간 객체가 추가됐을 때는 m.size가 19로 증가하였지만 두 번째 추가 시에는 카운터만 2로 증가하였고 m.size는 변하지 않았다. A에서처럼 릴레이션 저장소에서 삭제될 때 이전에 카운터가 2였기 때문에 카운터만 1로 줄어들었고 m.size가 줄어들지 않고 있다가(즉 해제가 되지 않고 있다가) B의 마지막에서처럼 윈도우 저장소에서 삭제될 때 카운터가 0이 되어서 m.size가 줄어들었다.



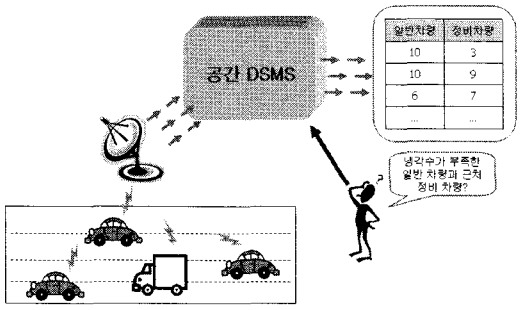
<그림 6> 공간 객체 관리자의 처리 기록

4. 실험 및 결과

본 장에서는 공간 DSMS가 활용될 수 있는 가상의 시나리오를 설정하고 시뮬레이션 함으로써 본 시스템의 유용성을 확인하였다. 이를 위하여 가상 시나리오에 적합한 질의문을 작성하고, 스트림 및 릴레이션 스키마를 정의하였으며, 또한 이동 센서를 위한 데이타 생성기를 이용하여 시뮬레이션에 필요한 가상 데이타를 생성하였다.

4.1 가상 시나리오

본 실험을 위하여 다음과 같은 가상 시나리오를 정의하였다. 그림 7에서와 같이 많은 일반 차량들과 정비 차량들이 고속도로를 달리고 있다. 정비 업체는 일반 차량들과 정비 차량들의 센서로부터 아이디와 위치를 수집하며, 일반 차량은 센서가 측정한 냉각수 잔량을 수집한다. 이러한 정보는 데이타 스트림 형태로 공간 DSMS에 입력된다. 정비 업체는 공간 DSMS에 냉각수 잔량이 3리터 이하인 일반 차량이 있을 경우 해당 일반 차량의 아이디를 근처 0.5km이내에 있는 정비 차량의 아이디와 함께 보고하도록 연속 질의를 등록함으로써 실시간으로 냉각수가 떨어진 차량에 대하여 정비 서비스를 수행할 수 있도록 한다.



<그림 7> 공간 DSMS 활용 예

4.2 실험에 사용된 데이타 및 질의문

표 4는 본 가상 시나리오에서 사용한 테이블 목록을 보여준다. 여기서 스트림과 릴레이션은 앞서 언급한 스트림 모델과 릴레이션 모델을 나타낸다. t는 샘플링 시간을 의미하며, loc은 위치를 WKT 형태의 문자열로 표현한 것이고, water는 냉각수 잔량을 의미한다. R은 일반 차량의 센서 데이타를 위한 테이블이며, S는 정비 차량의 센서 데이타를 위한 테이블이다. s_t, water_t, dist_t는 각각 질의 1, 질의 2, 질의 3의 결과가 저장되는 테이블이다. g는 loc으로부터 변환된 공간 객체를 의미하며, s_id와 r_id는 각각 S의 id와 R의 id를 의미하고, dist는 두 차량 간의 거리를 의미한다.

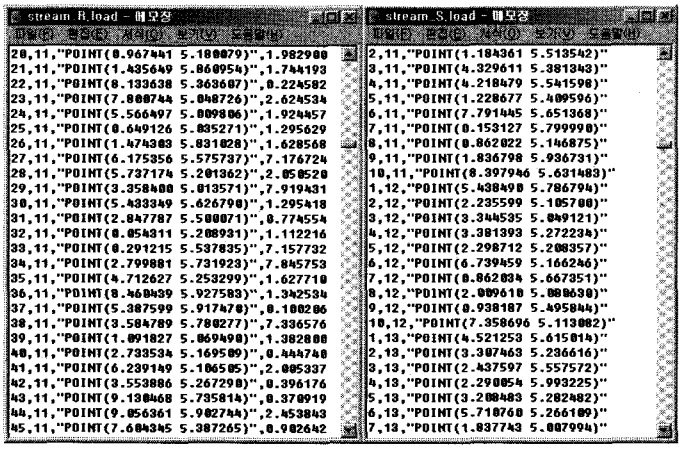
<표 4> 스트림과 릴레이션 테이블 목록

이 름	타 입	속 성
R	스트림	id Int, t Int, loc Char(30), water Int
S	스트림	id Int, t Int, loc Char(30)
s_t	릴레이션	id Int, t Int, g Geometry
water_t	릴레이션	id Int, t Int, g Geometry
dist_t	릴레이션	s_id Int, r_id Int, dist Float

본 시스템을 실험하기 위해서는 공간 데이타와 비공간 데이타를 동시에 생성할 수 있어야 하기 때문에 이에 적합한 이동 센서를 위한 데이타 생성기 [4]를 사용하였다. 이 생성기는 이동 센서의 특징을 고려하면서 공간 데이타와 비공간 데이타를 동시에 생성하기 위해 개발한 데이타 생성기로서 서너가지 파라미터를 입력해서 일반적으로 사용되는 정규분포나 비정규분포 혹은 균등분포를 따르는 데이타를 생성할 수 있다.

그림 8은 본 실험에 사용된 데이타를 보여준다. 그림 8(a)에 보이는 데이타는 일반 차량의 센서 데이타로써 (아이디, 샘플링 시간, 위치, 냉각수 잔량)으로 구성되며, 그림 8(b)에 보이는 데이타는 정비 차량의 센서 데이타로써 (아이디, 샘플링 시간, 위치)로 구성된다.

그림 9는 본 실험에서 설정한 시나리오에 맞는 결과를 획득하기 위해 사용된 질의문들을 보여준다. 여기서 질의 1은 "최근 2초 동안 S에 입력된 정비 차량 센서 데이타 중 아이디, 샘플링 시간, 객체로



(a) 일반 차량의 센서 데이타 (b) 정비 차량의 센서 데이타

<그림 8> 실험에 사용된 센서 데이타

변환된 위치를 출력하라."는 질의문이며, 질의 2는 "최근 2초 동안 R에 입력된 일반 차량 센서 데이타 중 냉각수 잔량이 3리터 미만인 차량들의 아이디, 샘플링 시간, 객체로 변환된 위치를 출력하라."는 질의이다. 질의 3은 "s_t와 water_t로부터 같은 샘플링 시간을 갖는 튜플들의 정비 차량 아이디, 일반 차량 아이디, 두 차량 간의 거리를 출력하라."는 질의이고, 질의 4는 "질의 3의 결과 중 거리가 0.5km 미만인 튜플들의 정비 차량 아이디와 일반 차량 아이디를 출력하라."는 질의이다.

질의 1 (s_t)	Select id, t, GeomFromText(S.loc, 0) From S [range 2 seconds];
질의 2 (water_t)	Select id, t, GeomFromText(R.loc, 0) From R [range 2 seconds] Where R.water < 3.0;
질의 3 (dist_t)	Select s_t.id, water_t.id, Distance(s_t.g, water_t.g) From s_t, water_t Where s_t.t = water_t.t;
질의 4	Select s_id, r_id From dist_t Where dist < 0.5;

<그림 9> 질의문 목록

질의 1과 질의 2에 있는 GeomFromText()는 WKT 형태의 문자열로부터 공간 객체를 Heap 영역에 생성하는 함수이고, 두 질의에 있는 [range 2 seconds]라는 조건을 통해 입력 S로부터 최근 2초간의 데이타만을 저장소에 저장하게 되고 2초가 지난 데이타는 저장소에서 삭제된다. 질의 3은 질의 1과 질의 2의 결과를 조인하는 질의이며, 저장소의 s_t.g 컬럼과 water_t.g 컬럼에 저장된 공간 객체의 id를 Distance() 함수의 인자로 입력하여 두 공간 객체의 거리 계산을 수행한다. 마지막으로 질의 4는 계산한 거리가 0.3km 미만인 튜플의 id를 출력한다.

그림 10은 질의 4를 만족하는 정비 차량 아이디(s_id)와 일반 차량 아이디(r_id)를 출력한 화면을 보여준다. 그림 9에 있는 질의 4번의 결과를 보면 현재 시스템 동작 후 11초가 지난 시점에서 정비 차량 2번으로부터 500미터 이내에 있으면서 냉각수가 3리터 미만인 일반 차량으로 20, 21, 26, 39번 차량이 검색되었다. 앞의 그림 8에 있는 데이타를 보면, 왼쪽 데이타의 첫줄에 있는 일반 차량 20번은 위치와 냉각수 잔량이 각각 "POINT (0.967441 5.180079)"와 1.982900이고, 오른쪽 데이타의 첫줄에 있는 정비 차량 2번의 위치가 "POINT(1.184361 5.513542)"이다. 이때 두 차량의 거리는 약 0.39이므로 결과가 정확하다. 이외에 다양한 연속 공간 질의를 실험한 결과 모두 정확한 결과가 나왔음을 확인할 수 있었다.

Query 4

Timestamp	Ins/Del	s_id	r_id
11	+	2	39
11	+	5	39
11	+	8	39
11	+	10	36
11	+	3	35
11	+	1	12
11	+	1	13
11	+	10	8
11	+	10	22
11	+	9	11
11	+	9	21
11	+	9	25
11	+	8	20
11	+	8	25
11	+	7	18
11	+	6	22
11	+	5	20
11	+	4	17
11	+	3	17
11	+	2	20
11	+	2	21
11	+	2	26
11	+	7	10
11	+	6	10
11	+	5	18

<그림 10> 질의 4의 결과 화면

이와 같이 공간 DSMS가 지속적으로 입력되는 공간 센서 데이타에 대한 공간 함수를 실시간으로 처리할 수 있음을 보여줌으로써 본 시스템이 실시간 공간 센서 데이타 모니터링 및 필터링이 필요한 환경에서 유용하게 사용될 수 있음을 알 수 있다.

5. 결론 및 향후 연구 과제

최근 유비쿼터스 환경에서는 공간 정보를 포함한 다양한 센서 데이타에 대한 실시간 처리 요구가 증가된다. 그러나 기존 DBMS는 실시간 모니터링을 하기 위해서는 질의를 일정 주기로 반복 수행해야 하는 문제가 있고, 또한 기존 DSMS는 공간 함수들을 지원하지 않는 문제가 있어서 이에 대한 보완이 필요하다.

따라서 본 논문에서는 연속 질의 처리를 위한 DSMS인 스탠포드 대학의 STREAM에 대하여 분석하고, 여기에 공간 데이타 스트림을 처리하기 위하여 OGC에서 제시한 Simple Feature Specification을 따르는 12가지 공간 객체와 18가지 공간 함수들을 처리할 수 있도록 STREAM을 확장하였다. 이를 위해서 내장 타입으로써 다양한 형태의 공간 데이타를 다룰 수 있는 공간 객체 타입, GEOS와 연동하여 공간 함수를 처리하기 위한 공간 함수 처리기, 여러 저장소들이 공유하는 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리자 등을 개발하였다. 그리고 공간 DSMS를 실시간 모니터링이 필요한 가상 시나리오에 적용하여 그 유용성을 검증하였다.

실시간 공간 센서 데이타를 효율적으로 인덱싱하기 위해서는 기존 공간 데이타베이스에서 사용하던 인덱싱 방식을 사용하기에는 인덱스 재구성 시간에 의한 비용이 너무 크기 때문에 앞으로 공간 DSMS에서 연속 질의를 효율적으로 처리하기 위해 인덱싱 기법에 대한 연구가 필요하겠다.

참고문헌

1. Steere, C., Baptista, A., McNamee, D., Pu, C., and Walpole, J., "Research Challenges in Environmental Observation and Forecasting Systems," Proc. of the International Conference on Mobile Computing and Networking, 2000, pp.292-299.
2. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J., "Wireless Sensor Networks for Habitat Monitoring," Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp.88-97.
3. 전상훈, 홍동숙, 김동오, 김정준, 한기준, "텔레매틱스를 위한 센서 데이타 서비스 시스템 개발," 한국공간정보시스템학회 추계학술대회논문집, 31권2호, 2005, pp.141-146.
4. 박치민, 김동오, 홍동숙, 한기준, "이동 센서를 위한 데이타셋 생성기," 한국GIS학회 GIS/RS 공동 춘계학술대회논문집, 2006, pp.131-137.
5. Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J., "Models and Issues in Data Stream Systems," Proc. of the ACM Symposium on Principles of Database Systems, 2002, pp.1-16.
6. Golab, L., and Özsu, M. T., "Issues in Data Stream Management," ACM SIGMOD

Record, Vol.32, No.2, 2003, pp.5-14.
7. Chen, J., DeWitt, D. J., Tian, F., and Wang, Y., "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," Proc. of the ACM SIGMOD International Conference on Management of Data, Vol.29, No.2, 2000, pp.379-390.
8. Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S., "Aurora: A New Model and Architecture for Data Stream Management," VLDB Journal, Vol.12, No.2, 2003, pp.120-139.
9. Cranor, C. D., Johnson, T., Spatscheck, O., and Shkapenyuk, V., "Gigascope: A Stream Database for Network Applications," Proc. of the ACM SIGMOD International Conference on Management of Data, 2003, pp.647-651.
10. Krishnamurthy, S., Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Madden, S. R., Reiss, F., and Shah, M. A., "TelegraphCQ: An Architectural Status Report," IEEE Data Engineering Bulletin, Vol.26, No.1, 2003, pp.11-18.
11. 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.
12. Hammad, M. A., Mokbel, M. F., Ali, M. H., Aref, W. G., Catlin, A. C., Elmagarmid, A. K., Eltabakh, M., Elfeky, M. G., Ghanem, T. M., Gwadera, R., Ilyas, I. F., Marzouk, M. S., and Xiong, X., "Nile: A Query Processing Engine for Data Streams," Proc. of the International Conference on Data Engineering, 2004, pp.851.
13. Open Geospatial Consortium Inc., *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option*, 2005.
14. Ramsey, P., *The State of Open Source GIS*, Refractions Research Inc., 2006.
15. Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J., and Widom, J., "STREAM: The Stanford Stream Data Manager," IEEE Data Engineering Bulletin, Vol.26, No.1, 2003, pp.19-26.
16. Arasu, A., Babu, S., and Widom, J., "The CQL Continuous Query Language: Semantic Foundations and Query Execution," VLDB Journal, Vol.15, No.2, 2006, pp.121-142.

강홍구
2002년 건국대학교 컴퓨터공학과(공학사)
2004년 건국대학교 대학원 컴퓨터공학과(공학석사)
2004년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야: 공간 데이타베이스, GIS, LBS, USN, 센서 데이타베이스

박치민
2004년 건국대학교 컴퓨터공학과(공학사)
2007년 건국대학교 대학원 컴퓨터공학과(공학석사)
2007년~현재 (주)다음커뮤니케이션 사원
관심분야: 공간 데이타베이스, GIS, LBS, 스트림 데이타베이스

홍동숙
1999년 건국대학교 컴퓨터공학과(공학사)
2001년 건국대학교 대학원 컴퓨터공학과(공학석사)
2000년~2003년 쌍용정보통신 모바일/GIS 기술팀
2003년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야: 데이타베이스, 이동체 데이타베이스, 유비쿼터스 컴퓨팅

한기준

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