대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어의 개발†

The Development of a Spatial Middleware for Efficient Retrieval of Mass Spatial Data

이기영* / Ki-Young Lee, **김동오**** / Dong-Oh Kim,
신중수*** / Jung-Su Shin, **한기준****** / Ki-Joon Han

요 약

최근 GIS 분야에서 공간 분석이나 군사적 목적을 위해 좀 더 상세하거나 넓은 영역의 공간 데이타를 요구함에 따라 대용량 공간 데이타의 효율적인 검색에 대한 필요성이 증대되고 있다. 이러한 분야에서 공간 데이타 검색에 활용되고 있는 Oracle Spatial이나 ESRI ArcSDE와 같은 기존의 GIS 소프트웨어는 공간 데이타의 안정적인 관리와 다양한 서비스를 위해 많은 기능들을 제공하지만, 저장 구조 및 공간 연산의 복잡성으로 인해 대용량 공간 데이타에 대한 검색이 비효율적이다.

따라서, 본 논문에서는 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어를 개발하였다. 본 논문에서 개발한 공간 미들웨어는 안정적인 공간 데이타의 관리를 위해 상용 DBMS인 Oracle을 데이타 저장소로써 활용하였으며, Oracle에 저장된 대용량 공간 데이타의 효율적인 접근을 위해 OCCI(Oracle C++ Call Interface)를 이용하였다. 또한, 대용량 공간 데이타의 효율적 연산과 검색을 위해 다양한 공간 연산 기법 및 Array Fetch 기법 등을 사용하였다. 마지막으로, 본 논문에서 개발한 공간 미들웨어와 Oracle Spatial 및 ESRI ArcSDE의 검색 성능을 비교함으로써 공간 미들웨어에 대용량 공간 데이타 검색 성능의 우수함을 입증하였다.

Abstract

Recently, because of the need to wide-area spatial data for spatial analysis and military purpose, there are increasing demand for the efficient retrieval of mass spatial data in Geographic Information System(GIS) fields. Oracle Spatial and ESRI ArcSDE, that are GIS Software, are to manage mass spatial data stably and to support various services but they are inefficient to retrieve mass spatial data because of the complexity of their spatial data models and spatial operations.

† 본 연구는 건설교통부 첨단도시기술개발사업-지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었음.

■ 논문접수 : 2007.10.11 ■ 심사완료 : 2008.1.10

* 을지대학교 의료산업학부 교수(kylee@eulji.ac.kr)
** 교신저자 건국대학교 컴퓨터공학부 강의교수(dokim@db.konkuk.ac.kr)
*** 건국대학교 대학원 컴퓨터 · 정보통신공학과 공학석사(bluenoel@gmail.com)
**** 건국대학교 컴퓨터공학부 교수(kjhan@db.konkuk.ac.kr)

Therefore, in this paper, we developed a spatial middleware that can retrieve mass spatial data efficiently. The spatial middleware used Oracle which is a representative commercial DBMS as a repository for the stable management of spatial data and utilized OCCI(Oracle C++ Call Interface) for the efficient access of mass spatial data in Oracle. In addition, various spatial operating methods and the Array Fetch method were used in the spatial middleware to perform efficient spatial operations and retrieval of mass spatial data in Oracle, respectively. Finally, by comparing the spatial middleware with Oracle Spatial and ESRI ArcSDE through the performance evaluation, we proved its excellent retrieval and storage performance of mass spatial data.

주요어 : GIS, 공간 미들웨어, 공간 관계 연산자, Oracle Spatial, ArcSDE
Keyword : GIS, Spatial Middleware, Spatial Relational Operator, Oracle Spatial, ArcSDE

1. 서 론

최근 GPS(Global Positioning System), LIDAR(Laser Imaging Detection And Ranging) 등과 같은 위치 측위 기술의 획기적인 발전으로 대용량의 공간 데이타가 축적되고, 정보 사회로 진화함에 따라 모바일 컴퓨팅, 텔레매틱스 등과 같은 다양한 분야에서 공간 데이타의 활용이 늘어나게 되었다. 특히, 지리 분야나 군사 분야와 같은 특정 분야에서는 좀 더 상세하거나 넓은 영역의 공간 데이타가 요구됨에 따라 대용량 공간 데이타의 효율적인 검색에 대한 필요성이 증대되고 있다[1].

이러한 환경에서 대용량의 공간 데이타를 서비스하기 위해 기반 시스템으로 Oracle Spatial[2,3]이나 ArcSDE[4,5]와 같은 소프트웨어가 활용되고 있다. 그러나 Oracle Spatial이나 ArcSDE와 같은 기존의 GIS 소프트웨어는 공간 데이타의 안정적인 관리와 다양한 서비스를 위해 많은 기능들을 제공하지만, 저장 구조의 복잡성과 기능의 다양성으로 인해 대용량 공간 데이타에 대한 검색이 비효율적이다[6,7]. 그러므로 이러한 환경에서 대용량 공간 데이타를 효율적으로 검색할 수 있는 기반 시스템이 절실히 필요한 실정이다.

따라서, 본 논문에서는 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어를 개발하였다. 공간 미들웨어는 안정적인 공간 데이타의 관리를 위해 상용 DBMS인 Oracle을 데이타 저장소로써 활용하며, OCCI(Oracle C++ Call Interface)[8]를 이용해서 Oracle에 저장된 대용량 공간 데이타를 저장, 검색, 갱신한다. 또한, Oracle로부터 대용량 공간 데이타를 효율적으로 가져오기 위해 Oracle로부터 정해진 개수만큼의 데이타를 한꺼번에 가져올 수 있는 Array Fetch 방식을 사용하며, 대용량 공간 데이타에 대한 공간 연산을 효율적으로 처리하기 위해 MBR(Minimum Bounding Rectangle) 연산[9], 폴리곤 내부(Interior) 연산[10], 공간 객체 연산[11]을 지원한다.

마지막으로, 본 논문에서 개발한 공간 미들웨어의 성능을 최적화하기 위한 Array Fetch 및 폴리곤 내부 연산의 설정에 대해서 분석하고, 본 논문에서 개발한 공간 미들웨어와 Oracle Spatial 및 ArcSDE 간의 검색 성능과 메모리 성능을 비교함으로써 공간 미들웨어가 대용량 공간 데이타의 검색과 메모리 사용 측면에서 Oracle Spatial 및 ArcSDE 보다 성능이 우수함을 입증하였다.

본 논문의 구성은 다음과 같다. 제 1 장 서론에 이어, 제 2 장에서는 관련 연구로 공간 미들웨어와 성능 비교 대상이 되는 Oracle Spatial과 ArcSDE에 대해 분석한다. 제 3 장에서는 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어에 대해 상세히 설명하고, 제 4 장에서는 성능 평가 환경과 다양한 성능 평가의 결과에 대해 설명한다. 마지막

```
CREATE TYPE SDO_GEOMETRY AS OBJECT (
   SDO_GTYPE number,    SDO_SRID number,
   SDO_POINT SDO_POINT_TYPE,
   SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
   SDO_ORDINATES SDO_ORDINATE_ARRAY);
CREATE TYPE SDO_POINT_TYPE AS OBJECT (
  X number,   Y number,   Z number);
CREATE TYPE SDO_ELEM_INFO_ARRAY AS varray(1048576) of number;
CREATE TYPE SDO_ORDINATE_ARRAY AS varray(1048576) of number;
```

<그림 1> 공간 데이타 타입 정의

으로, 제 5 장에서는 결론을 기술한다.

2. 관련연구

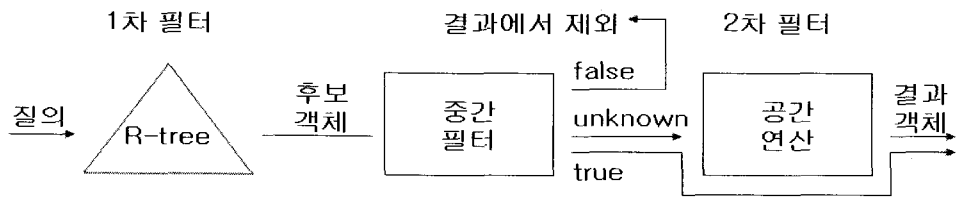
본 장에서는 관련 연구로써 본 논문에서 개발한 공간 미들웨어와 성능 비교 대상이 되는 Oracle Spatial과 ArcSDE에 대해 분석한다.

2.1 Oracle Spatial

Oracle Spatial은 Oracle에서 공간 데이타를 효율적으로 저장, 접근, 분석할 수 있도록 제공되는 함수와 프로시저의 집합이다[2,3]. Oracle Spatial에서 공간 데이타는 SDO_GEOMETRY 타입을 이용해 테이블에 저장된다. <그림 1>는 Oracle Spatial에서 사용되는 공간 데이타 타입의 정의를 보여준다.

<그림 1>에서 보듯이 SDO_GEOMETRY는 공간 데이타 타입의 종류를 저장하기 위한 SDO_GTYPE, 좌표계의 종류를 저장하기 위한 SDO_SRID, 단일 좌표를 저장하기 위한 SDO_POINT, SDO_ORDINATES의 구성 정보를 저장하기 위한 SDO_ELEM_INFO, 다중 좌표를 저장하기 위한 SDO_ORDINATES로 구성된다. <그림 2>는 Oracle Spatial의 공간 연산 처리 과정을 보여준다.

<그림 2>에서 보듯이 Oracle Spatial의 공간 연산 처리 과정은 1차 필터, 중간 필터, 2차 필터의 단계로 이루어진다. 1차 필터와 중간 필터는 2차 필터 과정의 수행 전에 필터링을 수행해 공간 연산이 필요한 대상 공간 객체의 개수를 줄이는 과정이다. 1차 필터는 R-tree와 같은 공간 인덱스를 이용해 대상 공간 객체를 필터링하며, 중간 필터는 질의 공간 객체가 폴리곤일 경우 질의 폴리곤의 내부와 대상 공간 객체 MBR의 위상 관계를 이용하여 대상 공간 객체를 필터링한다[4,10]. 마지막으로 2차 필터는 실제 대상 공간 객체에 대해 공간 연산을 수행한다.



<그림 2> Oracle Spatial의 공간 연산 처리 과정

2.2 ArcSDE

ArcSDE는 상용 DBMS(예를 들어, Microsoft SQL Server, Oracle 등)의 GIS 게이트웨이로써 공간 데이타에 대한 저장, 검색, 관리를 지원하는 소프트웨어이다[4]. 또한, ArcSDE는 네트워크와 인터넷 상에서 동시에 여러 사용자를 대상으로 서비스 및 협력 작업을 지원한다. 본 논문에서 비교 대상인 Oracle 기반의 ArcSDE는 공간 데이타를 Oracle의 long row나 blob(Binary Large Object) 타입으로 저장한다. <그림 3>은 ArcSDE에서 공간 데이타를 저장하는 Feature 테이블의 구조를 보여준다.

F<layer_id>	
fID	number(38)
numofPTs	number(38)
entity	number(38)
eMinX	number(64)
eMinY	number(64)
eMaxX	number(64)
eMaxY	number(64)
eMinZ	number(64)
eMaxZ	number(64)
area	number(64)
len	number(64)
points	long row

<그림 3> Feature 테이블의 구조

<그림 3>에서 보듯이 Feature 테이블의 구조는 객체의 식별자인 fID, 객체를 구성하는 포인트(Point)의 개수인 numofPTs, 공간 객체의 타입 정보인 entity, 3차원 MBR 정보인 eMinX, eMinY, eMaxX, eMaxY, eMinZ, eMaxZ, 넓이 및 길이 값인 area와 len, 공간 객체를 구성하는 포인트의 바이트 스트림인 points로 구성된다.

ArcSDE는 공간 인덱스로써 Multi-level Grid 인덱스를 지원한다. <그림 4>는 ArcSDE에서 제공하는 Multi-level Grid 인덱스의 구성을 보여준다.

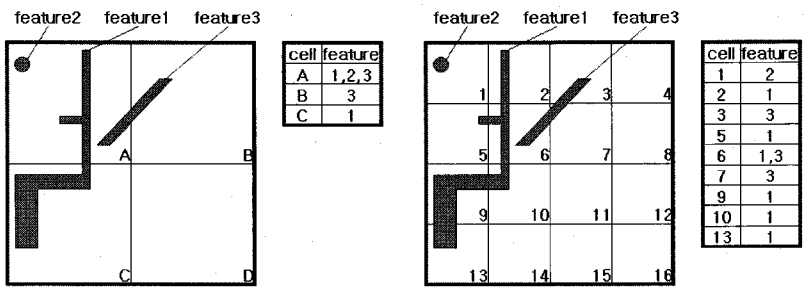
<그림 4>에서 보듯이 Multi-level Grid 인덱스는 각 Grid 레벨별로 각 셀에 존재하는 공간 객체의 정보를 가진다. Multi-level Grid에서 검색 시 질의 영역이 특정 레벨 Grid의 셀을 포함한다면 해당 셀의 하위 레벨 Grid의 셀은 더 이상 검색할 필요가 없다. 즉, 질의 영역이 <그림 4(a)>에서 셀 A를 모두 포함한다면 해당 셀의 하위 레벨 Grid인 <그림 4(b)>의 셀 1, 2, 5, 6은 더 이상 검색할 필요가 없다.

3. 공간 미들웨어

본 장에서는 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어의 전체 시스템 구조, 내부 저장 데이타 구조, 각 세부 관리자에 대해서 상세하게 설명한다.

3.1 전체 시스템 구조

대용량의 공간 데이타를 서비스하기 위해 기반

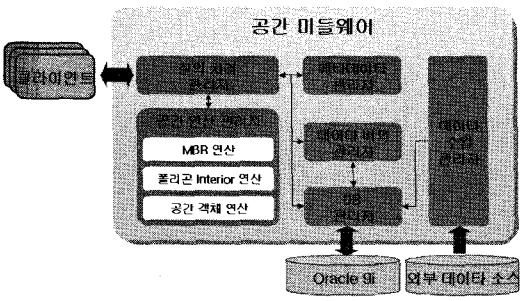


(a) Index Grid 1 (b) Index Grid 2

<그림 4> Multi-level Grid 인덱스 구성

시스템으로 사용되는 Oracle Spatial이나 ArcSDE는 저장 구조의 복잡성과 기능의 다양성으로 인해 대용량 공간 데이타에 대한 검색이 비효율적이다. 특히, Oracle Spatial은 소수의 공간 객체 검색에는 매우 빠르나 객체 형태의 공간 데이타 타입을 활용함으로써 Array Fetch 방식을 적용할 수 없으며, ArcSDE는 Array Fetch 방식을 적용 할 수는 있으나, 메모리 소모가 많고 공간 연산 필터 처리가 부족하여 대용량 공간 데이타 처리가 효율적이지 못하다.

이를 해결하기 위해 본 논문에서는 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어를 개발하였다. 공간 미들웨어는 공간 데이타의 관리를 위해 자체적인 데이타 구조에 따라 안정적인 상용 DBMS인 Oracle을 데이타 저장소로써 활용하며, OCCI를 이용해서 Oracle에 저장된 대용량 공간 데이타를 저장, 검색, 갱신한다. 특히, 효율적인 대용량 공간 데이타에 대한 공간 연산을 위해 MBR 연산, 폴리곤 내부(Interior) 연산, 공간 객체 연산을 지원한다. 또한, 자체적인 데이타 구조를 사용함으로써 대용량 공간 데이타에 대한 전송을 효율적으로 처리하기 위해 Array Fetch 방식이 사용 가능하다. 본 논문에서 개발한 공간 미들웨어의 전체 시스템 구조는 <그림 5>와 같다.

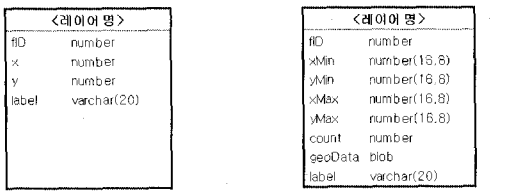


<그림 5> 전체 시스템 구조

<그림 5>에서 보듯이 공간 미들웨어는 클라이언트에서 요청하는 질의를 분석 및 처리하기 위한 질의 처리 관리자, 공간 데이타를 Oracle에서 효율적으로 검색 및 관리하기 위한 DB 관리자, 공간 데이타의 효율적인 공간 연산을 위한 MBR 연산, 폴리곤 내부 연산, 공간 객체 연산을 지원하기 위한 공간 연산 관리자, 자주 사용되는 공간 데이타를 공간 미들웨어의 메모리에서 관리함으로써 빠른 검색을 지원하기 위한 데이타 버퍼 관리자, 외부 데이타 소스로부터 대용량의 공간 데이타를 추출 및 변환해 Oracle에 저장하기 위한 데이타 수입 관리자, 공간 미들웨어에서 검색 및 관리 시 사용되는 다양한 메타데이타를 관리하기 위한 메타데이타 관리자로 구성되어 있다.

3.2 내부 저장 데이타 구조

공간 미들웨어는 공간 데이타의 안정적 관리를 위해 내부 데이타 저장소로써 Oracle을 사용한다. Oracle에서 공간 데이타는 효율적으로 관리되기 위해 공간 데이타 타입별로 정해진 스키마에 따라 저장된다. 또한, 각 레이어별로 해당 공간 데이타 타입에 맞는 별도의 테이블을 생성한다. <그림 6>은 공간 데이타 타입별로 각 레이어를 저장하기 위한 구조를 보여준다.



(a) 포인트 타입 레이어 구조 (b) 라인스트링 및 폴리곤 타입 레이어 구조

<그림 6> 공간 데이타 타입별 레이어 구조

<그림 6>에서 보듯이 각 레이어 별로 공간 데이타 타입에 따라 해당 레이어의 이름을 가지는 하나의 테이블이 생성된다. <그림 6(a)>는 포인트 타입 레이어를 저장하기 위한 구조로써, 공간 객체의 식별자인 fID, 포인트의 좌표인 x와 y, 해당 공간 객체의 간단한 설명인 label을 가진다. <그림 6(b)>는 라인스트링(LineString) 및 폴리곤(Polygon) 타입 레이어를 저장하기 위한 구조로써, 공간 객체의 식별자인 fID, MBR인 xMin, yMin, xMax,

<표 1> 질의 처리 관리자의 주요 함수

함 수	기 능
int GetResultMetaQuery (byte * qResult);	메타데이타 검색 질의 처리
int GetResultMBRQuery (byte * qResult);	MBR 공간 영역 검색 질의 처리
int GetResultObjectQuery (byte * qResult);	객체 공간 영역 검색 질의 처리
int GetResultInsertDML (byte * qResult);	삽입 DML문 처리
int GetResultUpdateDML (byte * qResult);	갱신 DML문 처리
int GetResultDeleteDML (byte * qResult);	삭제 DML문 처리
int GetResultAttQuery (byte * qResult);	객체 속성 검색 질의 처리

yMax와 해당 공간 객체를 구성하는 좌표의 개수인 count를 가지며 좌표들을 저장하기 위해 blob 타입의 geoData와 해당 공간 객체의 간단한 설명인 label을 가진다.

3.3 질의 처리 관리자

질의 처리 관리자는 클라이언트로부터 요청된 공간 및 비공간 데이타의 검색, 삽입, 삭제, 갱신 질의를 처리하기 위한 관리자이다. 질의 처리 관리자는 메타데이타 관리자, 공간 연산 관리자, 데이타 버퍼 관리자, DB 관리자와 연동해서 요청된 질의를 처리한다. 질의 처리 관리자는 클라이언트의 질의 요청 시 구문 분석 과정을 통해 메타데이타 관리자와 연동해서 질의문의 오류를 판별하거나 해당 레이어의 데이타 버퍼 정보 등을 파악한다. 그리고 질의 처리 관리자는 데이타 버퍼 관리자 또는 DB 관리자를 통해 검색이 필요한 공간 데이타에 접근한 후 공간 연산 관리자에 공간 연산의 처리를 요청한다. <표 1>은 질의 처리 관리자의 주요 기능을 수행하는 함수를 보여준다.

<표 1>에서 GetResultMetaQuery 함수는 메타데이타 정보를 획득하기 위해 메타데이타 관리자와 연동하기 위한 함수이며, GetResultMBRQuery, GetResultObjectQuery 함수는 MBR 연산 및 공간 객체 연산을 위해 공간 연산 관리자와 연동하기 위한 함수이다. 그리고 GetResultInsertDML, GetResultUpdateDML, GetsultDeleteDML 함수는 데이타의 삽입, 갱신, 삭제를 처리하기 위한 함수이며, GetResultAttQuery 함수는 비공간 속성에 대한 검색 질의를 처리하기 위한 함수이다.

<표 2>는 공간 미들웨어에서 지원하는 공간 연산의 종류를 보여주며, 여기서 'X' 표시는 해당 공간 연산이 불가능함을 나타낸다.

<표 2> 공간 미들웨어에서 지원하는 공간 연산

공간 연산	질의 영역의 타입 / 대상 공간 객체 타입								
	Point/ Point	Point/ Line	Point/ Polygon	Line/ Point	Line/ Line	Line/ Polygon	Polygon /Point	Polygon /Line	Polygon /Polygon
Disjoint									
Intersects									
Equals		X	X	X		X	X	X	
Crosses	X								X
Touches	X								
Within				X			X	X	
Contains		X	X			X			
Overlaps	X	X	X	X		X	X	X	

$$t = \triangle Mx + \frac{R}{x}(F_{\min} + \triangle F(x-1))$$
$$= \triangle Mx + \frac{R(F_{\min} - \triangle F)}{x} + \triangle FR$$

계 수	설 명	측정값
$F_{\min}$	단일 데이타 Fetch 시간	0.264 (ms)
$\triangle F$	ARRAYFETCH가 1 증가할 때 단일 Array Fetch 시간의 증가분	0.015 (ms)
$\triangle M$	ARRAYFETCH가 1 증가할 때 메모리 할당 시간의 증가분	0.025 (ms)
R	Oracle로부터 Fetch할 전체 로우 개수	가변
x	ARRAYFETCH : Array Fetch시 한꺼번에 가져올 데이타 개수	
t	전체 Fetch 시간	

<그림 7> ARRAYFETCH에 따른 Fetch 수행 시간 함수

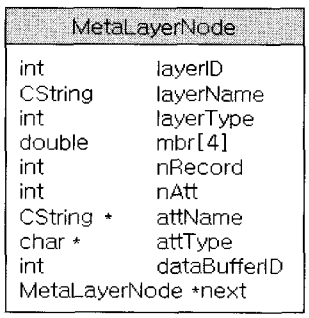
3.4 DB 관리자

DB 관리자는 Oracle에 접속 하여 대용량 데이타를 효율적으로 저장 및 검색하기 위한 관리자로써, Oracle에서 제공하는 OCCI를 이용해서 Oracle과 연결하여 저장, 검색, 갱신 요청을 처리한다. DB 관리자는 Oracle로부터 대용량 공간 데이타를 효율적으로 가져오기 위해 Array Fetch 방식을 사용한다. Array Fetch 방식은 데이타를 저장할 공간을 미리 할당하고 Oracle로부터 정해진 개수만큼의 데이타를 한꺼번에 가져온다. Array Fetch의 성능은 한꺼번에 가져올 데이타의 개수인 ARRAYFETCH에 따라 결정된다. <그림 7>은 Array Fetch 적용 시 ARRAYFETCH에 따른 전체 Fetch 시간에 대한 함수를 보여준다.

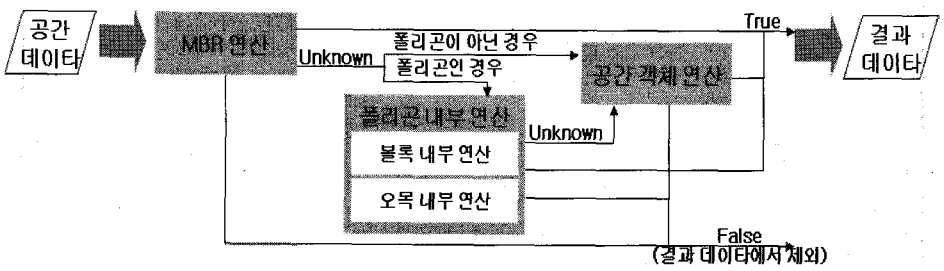
<그림 7>에서 보듯이 전체 Fetch 시간인 t는 $F_{\min}$, $\triangle F$, $\triangle M$, R, x에 의해서 결정되며, 이 중 $F_{\min}$, $\triangle F$, $\triangle M$는 공간 미들웨어에서 실험에 의해 구할 수 있는 값이고, R은 Oracle로부터 Fetch할 전체 로우의 개수, 즉 질의 결과의 개수이다. 따라서 공식에 이들을 대입함으로써 전체 Fetch 시간 t가 최소가 되는 x, 즉 ARRAYFETCH를 구할 수 있다.

3.5 메타데이타 관리자

메타데이타 관리자는 공간 미들웨어에서 검색 및 관리 시 사용되는 다양한 메타데이타를 관리하기 위한 관리자이다. 메타데이타는 레이어 정보와 데이타 버퍼 정보로 이루어진다. 메타데이타 관리자의 메타데이타는 공간 미들웨어 시작 시에 DB 관리자를 통해 Oracle에 저장된 메타데이타 테이블을 이용해 생성된다. 또한 메타데이타 관리자는 공간 미들웨어 구동 중에 데이타 수입 관리자를 통한 데이타 저장이나 클라이언트의 갱신 요청 등으로 인해 메타데이타의 갱신이 필요한 경우 DB 관리자를 통해 메타데이타 테이블을 갱신한다. <그림 8>은 메타데이타 저장을 위한 자료 구조를 보여준다.



<그림 8> 메타데이타의 자료 구조



<그림 9> 공간 연산 관리자의 공간 연산 처리 과정

<그림 8>의 MetaLayerNode는 연결 리스트 형태로 각 레이어의 메타데이타를 저장하기 위한 자료 구조로써 해당 레이어의 식별자인 layerID, 레이어의 이름을 나타내는 layerName, 레이어의 공간 데이타 타입인 layerType, 레이어의 전체 MBR인 mbr, 레이어에 포함된 데이타의 개수인 nRecord, 레이어의 속성 개수인 nAtt, 각 속성의 이름인 attName, 각 속성의 타입인 attType, 데이타 버퍼 관리자에서 해당 레이어의 데이타를 가진 데이타 버퍼의 식별자인 dataBufferID, 다음 레이어 정보를 가리키는 next를 가진다.

3.6 공간 연산 관리자

공간 연산 관리자는 효율적으로 공간 연산을 처리하기 위해 MBR 연산, 폴리곤 내부 연산, 공간 객체 연산을 수행하는 관리자이다. 공간 연산 관리자의 공간 연산 처리는 MBR 연산, 폴리곤 내부 연산, 공간 객체 연산의 순서로 처리되며, 선행 연산의 결과에 따라 후행 연산을 수행하지 않아도 되는 경우가 발생한다. <그림 9>는 공간 연산 관리자의 공간 연산 처리 과정을 보여준다.

<그림 9>에서 MBR 연산은 질의 영역의 MBR과 대상 공간 객체의 MBR 간에 위상 관계에 따라 공간 연산의 결과를 알 수 있는 경우 해당 공간 객체를 후행 연산 대상에서 제외한다. MBR 연산은 공간 객체 MBR의 좌표 속성을 이용하여 구성된 Oracle의 결합 인덱스를 이용하여 MBR 연산을 위해 구현된 함수를 통해 수행된다. <표 3>은 MBR 연산과 공간 연산의 관계를 보여준다.

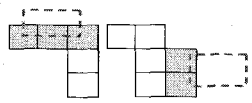
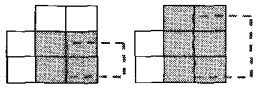
<표 3>에서 TRUE와 FALSE는 MBR 연산의 결과만으로 공간 연산의 결과가 TRUE나 FALSE임을 알 수 있는 경우를 보여주는데, 이러한 경우에는 결과를 얻기 위해 후행 공간 연산을 거치지 않아도 된다. 그러나 UNKNOWN은 공간 연산의 결과를 알기 위해서 후행 공간 연산을 계속해서 수행해야 한다.

<그림 9>에서 폴리곤 내부 연산은 질의 영역의 폴리곤 내부와 대상 공간 객체의 MBR 간에 위상 관계에 따라 공간 연산의 결과를 알 수 있는 경우

<표 3> MBR 연산 결과와 공간 연산의 관계

MBR 연산	공간 연산				
	Disjoint	Intersects	Equals	Within / Contains	Crosses / Touches / Overlaps
Disjoint	TRUE	FALSE	FALSE	FALSE	FALSE
Contains	UNKNOWN	UNKNOWN	FALSE	UNKNOWN	UNKNOWN
Overlaps	UNKNOWN	UNKNOWN	FALSE	FALSE	UNKNOWN

<표 4> 오목 폴리곤 내부와 MBR 간의 위상 관계에 따른 공간 연산의 관계

오목 폴리곤 내부와 MBR 간의 위상 관계	공간 연산				
	Contains	Intersects	Disjoint	Touches	Within / Crosses / Overlaps
	TRUE	UNKNOWN	UNKNOWN	UNKNOWN	FALSE
	UNKNOWN	TRUE	FALSE	UNKNOWN	UNKNOWN
	UNKNOWN	UNKNOWN	UNKNOWN	FALSE	UNKNOWN

해당 공간 객체를 후행 연산 대상에서 제외함으로써, 상대적으로 많은 비용이 발생하는 폴리곤 타입의 공간 연산이 효과적으로 처리될 수 있도록 한다. 폴리곤 내부 연산은 폴리곤의 종류에 따라 볼록(Convex) 내부 연산 또는 오목(Concave) 내부 연산을 수행한다.

볼록 내부 연산은 질의 영역이 볼록 폴리곤일 경우 질의 영역 내부의 가장 큰 사각형 영역과 대상 공간 객체 MBR 간에 위상 관계를 이용하여 후행 연산이 필요한 대상 공간 객체의 개수를 줄여준다. 예를 들어, 질의 영역의 볼록 폴리곤 내부가 대상 공간 객체의 MBR을 Contains하면 Overlaps/Disjoint/Touches/Within/Crosses 공간 연산의 결과는 무조건 FALSE가 되며, Intersects/Contains 공간 연산의 결과는 무조건 TRUE가 된다. 오목 내부 연산은 질의 영역이 오목 폴리곤일 경우 질의 영역을 그리드 형태로 분할한 각 영역 중에서 폴리곤에 완전히 포함되는 영역과 대상 공간 객체의 MBR 간에 위상 관계를 이용하여 후행 연산이 필요한 대상 공간 객체의 개수를 줄여준다. <표 4>는 질의 영역의 오목 폴리곤 내부와 대상 공간 객체의 MBR 간에 위상 관계에 따른 공간 연산의 관계를 보여준다.

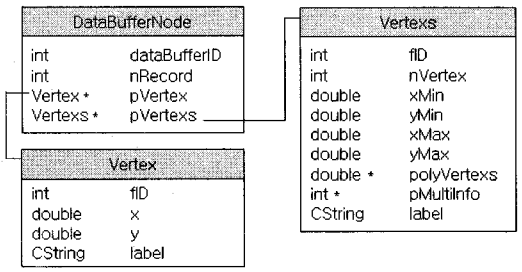
<표 4>에서 점선 사각형은 대상 공간 객체의 MBR이며 실선 사각형은 질의 영역의 오목 폴리곤 내부이다. 또한, TRUE와 FALSE는 오목 내부 연산의 결과만으로 공간 연산의 결과가 TRUE나 FALSE임을 알 수 있는 경우를 보여주는데, 이러한 경우에는 결과를 얻기 위해서 후행 공간 연산인 공간 객체 연산을 거치지 않아도 된다. 그러나 UNKNOWN은 공간 연산의 결과를 알기 위해서 후행 공간 연산을 계속해서 수행해야 한다.

마지막으로, 공간 연산 관리자는 MBR 연산, 폴리곤 내부 연산에서 대상 공간 객체의 개수를 충분히 줄인 뒤에 결과를 예측할 수 없는 공간 데이타에 대해서만 공간 객체 연산을 수행한다.

3.7 데이타 버퍼 관리자

데이타 버퍼 관리자는 공간 데이타에 대한 검색 시간을 단축하기 위해 자주 사용되는 레이어의 공간 데이타를 공간 미들웨어의 메모리 영역 상에 할당된 데이타 버퍼에 저장하고 검색하기 위한 관리자이다. 데이타 버퍼 관리자는 지정된 레이어의 공간 데이타를 DB 관리자를 통해 읽어 해당 데이타 버퍼에 저장하고 메타데이타 관리자를 통해 해당 레이어의 메타데이타에 데이타 버퍼 사용 정보를 기록한다. 이렇게 생성된 데이타 버퍼는 클라이언트의 질의에 대한 빠른 응답 시간을 보장한다. <그림 10>은 데이타 버퍼를 구성하는 데이타 버퍼 노

드의 자료 구조를 보여준다.



<그림 10> 데이타 버퍼 노드의 자료 구조

<그림 10>에서 DataBufferNode는 데이타 버퍼 노드를 저장하기 위한 자료 구조로써 메타데이타에서 참조되는 데이타 버퍼 식별자인 dataBufferID, 데이타 버퍼에 저장된 레코드 수인 nRecord, 포인트 타입의 공간 객체를 저장하기 위한 포인터인 pVertex, 라인스트링 및 폴리곤 타입의 공간 객체를 저장하기 위한 포인터인 pVertexs를 가진다. Vertex는 포인트 타입의 공간 객체를 저장하기 위한 자료 구조로써 공간 객체의 식별자인 fID, 포인트의 좌표인 x와 y, 해당 공간 객체의 간단한 설명인 label을 가진다. Vertexs는 라인스트링이나 폴리곤 타입의 공간 객체를 저장하기 위한 자료 구조로써 공간 객체의 식별자인 fID, 해당 공간 객체를 구성하는 좌표의 개수인 nVertex, MBR인 xMin, yMin, xMax, yMax, 공간 객체를 구성하는 좌표들인 polyVertexs, 해당 공간 객체의 구성 정보인 pMultiInfo, 해당 공간 객체의 간단한 설명인 label을 가진다.

3.8 공간 데이타 수입 관리자

공간 데이타 수입 관리자는 외부 데이타 소스를 분석해 공간 미들웨어에서 활용하기 위해 공간 데이타, 속성 데이타, 메타데이타를 추출 및 변환하여 Oracle에 저장하기 위한 관리자이다. <표 5>는 공간 미들웨어에서 외부 데이타 소스로 사용되는 Shape 화일 포맷의 구성을 보여준다.

<표 5> Shape 화일 포맷의 구성

화일유형	기 능
.shp	지리 현상의 기하학 정보를 저장하는 화일
.dbf	지리 현상의 속성 정보를 저장하는 포맷 화일

공간 데이타 수입 관리자는 Shape 화일을 Oracle 내부에 각 레이어 별 데이타 저장 구조에 따라 저장한다. Oracle 내부에 저장되는 공간 데이타는 .shp 화일을 기반으로 생성되며, 속성 데이타는 .dbf 화일을 기반으로 생성된다. 또한 메타데이타에 저장되는 레이어의 MBR, 레이어의 각 속성에 대한 데이타 타입 및 개수 등은 .shp 화일과 .dbf 화일의 헤더 정보를 기반으로 생성된다.

4. 성능 평가

본 장에서는 성능 평가를 위해 사용된 성능 평가 환경에 대해 설명한다. 그리고 공간 미들웨어에서 ARRAYFETCH와 오목 폴리곤 내부의 타일링 레벨에 따른 검색 성능을 평가한다. 마지막으로, 공간 미들웨어, Oracle Spatial, ArcSDE 간의 성능을 비교 평가한다.

4.1 성능 평가 환경

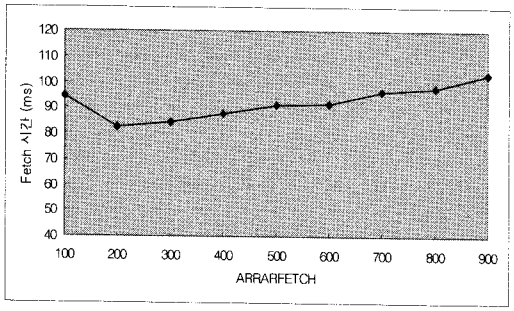
본 논문에서 성능 실험은 인텔 펜티엄 D 2.8GHz의 CPU와 1GB 메모리를 가진 Window XP 상에서 수행되었다. 또한, 공간 미들웨어를 구현하기 위한 개발 언어로는 Microsoft Visual C++ 7.0을 사용하였고, Oracle과의 연동을 위해 OCCI를 사용하였다. 성능 평가에 사용된 공간 데이타는 서울시의 건물 정보를 표현하는 폴리곤 타입의 객체를 사용하였다. 성능 평가에 사용한 공간 연산은 MBR Intersects, Contains, Intersects 이다.

공간 미들웨어와의 검색 성능 비교를 위해 Oracle Spatial 9.2.0.7과 ArcSDE 9.1 버전을 사용해 실험을 수행하였다. 공간 데이타의 구성은 공간 미들웨어에서는 공간 데이타 수입 관리자를 통해

자체적인 BLOB 타입으로 저장하였으며, Oracle Spatial에서는 shp2sdo를 통해 SDO_GEOMETRY 타입으로 저장하였고, ArcSDE에서는 ArcCatalog를 통해 ArcSDE Compressed Binary 타입으로 저장하였다. 또한 성능 평가 시 사용된 공간 인덱스는 공간 미들웨어에서는 결합 인덱스를 사용하였고, Oracle Spatial에서는 R-tree를 사용하였으며, ArcSDE에서는 Multi-level Grid 인덱스를 사용하였다.

4.2 ARRAYFETCH에 따른 검색 성능

본 논문의 DB 관리자는 Oracle로부터 대용량 공간 데이타를 효율적으로 가져오기 위해 Array Fetch 방식을 사용한다. Array Fetch 방식은 데이타를 저장할 공간을 미리 할당하고 Oracle로부터 정해진 개수만큼의 데이타를 한꺼번에 가져온다. Array Fetch의 성능은 한꺼번에 가져올 데이타의 개수인 ARRAYFETCH에 따라 결정된다. <그림 11>은 5,000개의 공간 데이타에 대해 Array Fetch 적용 시 ARRAYFETCH에 따른 Fetch 시간의 변화를 보여주는데, 이것은 <그림 7>에서 제시한 함수의 그래프와 유사한 결과를 보여준다.

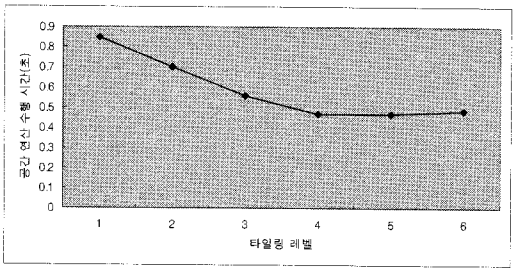


<그림 11> ARRAYFETCH에 따른 Fetch 시간

<그림 11>에서 보듯이 5,000개의 공간 데이타를 Fetch할 경우 ARRAYFETCH가 200일 때 최소의 Fetch 시간을 갖는다. 이처럼 ARRAYFETCH가 특정 값에서 좋은 성능을 내는 이유는 ARRAYFETCH가 증가할 경우 Oracle에 접근하기 위한 시간은 감소하지만, Array Fetch를 위한 메모리 할당 시간 및 단일 Fetch 시간이 증가함에 따라 Array Fetch로 인한 시간의 이득이 상대적으로 줄어들기 때문이다.

4.3 오목 폴리곤 내부의 타일링 레벨별 검색 성능

본 논문의 공간 연산 처리 과정에서 사용되는 MBR 연산 및 볼록 내부 연산 등은 하나의 영역을 이용해 비교 연산을 수행하지만, 오목 내부 연산은 타일링 레벨에 따라 하나 이상의 영역을 이용해 비교 연산을 수행하게 된다. 즉, 오목 내부 연산에서 비교 연산의 수행 대상이 되는 영역의 수는 오목 폴리곤 내부에 해당되는 타일의 개수이다. 이러한 타일의 개수는 타일을 나누는 횟수인 타일링 레벨과 비례한다. <그림 12>는 오목 폴리곤 내부의 타일링 레벨별 공간 연산의 성능을 보여준다.



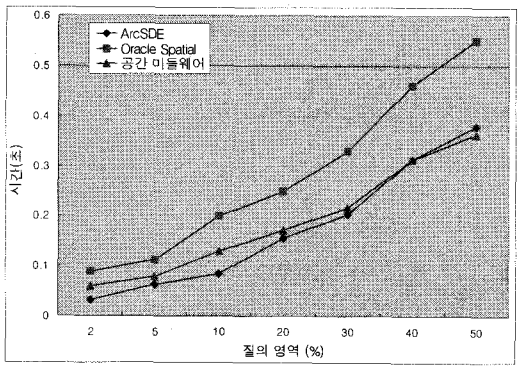
<그림 12> 오목 폴리곤 내부의 타일링 레벨별 공간 연산 성능

<그림 12>에서 오목 폴리곤 내부의 타일링 레벨이 4를 넘으면 성능의 개선이 보이지 않는다. 이는 타일링 레벨이 1레벨 증가할 때마다 오목 폴리곤 내부를 구성하는 타일의 수가 4배씩 증가하는데, 이러한 오목 폴리곤 내부에 포함되는 타일 개수의 증가로 인해 오목 내부 연산의 비교 횟수가 증가되기 때문이다.

4.4 MBR 검색 성능

MBR 검색의 성능 평가는 상대적으로 높은 공간

연산 비용을 보이는 폴리곤/폴리곤 연산 중 MBR Intersects 연산을 이용하였으며, 50,000개의 공간 데이타에 대해 질의 영역의 크기를 변화시키면서 공간 미들웨어, ArcSDE, Oracle Spatial의 검색 시간을 비교하였다. <그림 13>은 MBR Intersects 연산의 성능 평가 결과를 보여준다.



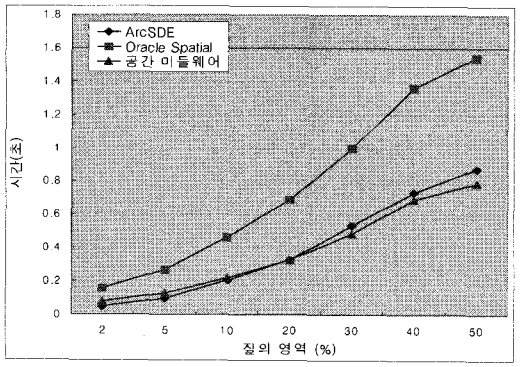
<그림 13> MBR Intersects 공간 연산 성능 비교

<그림 13>에서 보듯이 실험 결과를 통해 공간 미들웨어가 Oracle Spatial과 비교해서 평균 1.5배의 성능 향상을 보이며, ArcSDE와 비교해서 큰 차이는 없으나 질의 영역이 40% 이상으로 대용량 공간 데이타를 검색할 때 공간 미들웨어의 성능이 가장 빠름을 알 수 있다. 이를 통해 공간 미들웨어가 Array Fetch와 같은 방법을 통해 데이타베이스에 대한 접근 횟수를 줄임으로 인해 대용량 공간 데이타에 대한 검색 속도를 개선하였지만, 질의 영역이 작을 경우 ArcSDE의 Multi-level Grid 인덱스에 비해 인덱스 내부 연산 속도로 인해 검색 속도가 상대적으로 떨어짐을 알 수 있다.

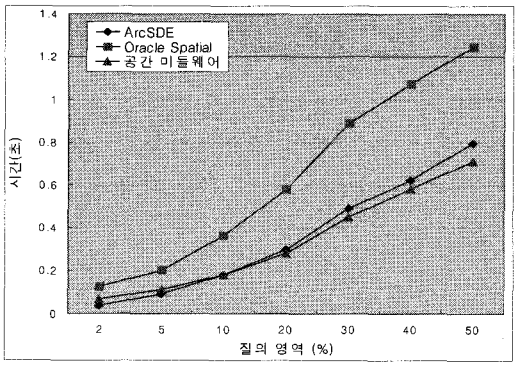
4.5 공간 객체 검색 성능

공간 객체 검색의 성능 평가는 상대적으로 높은 공간 연산 비용을 보이는 폴리곤/폴리곤 연산 중 Contains, Intersects 연산을 이용하였으며, 50,000개의 공간 데이타에 대해 질의 영역의 크기를 변화시키면서 공간 미들웨어, ArcSDE, Oracle Spatial의 검색 시간을 비교하였다. <그림 14>와 <그림 15>는 각각 Contains 연산과 Intersects 연산의 성능 평가 결과를 보여준다.

<그림 14>에서 보듯이 실험 결과를 통해 공간 미들웨어의 Contains 연산이 Oracle Spatial과 비교해서 평균 2배의 높은 성능 향상을 보이며, ArcSDE와 비교해서 질의 영역이 약 20% 이상으로 대용량 공간 데이타를 검색할 때 공간 미들웨어의 성능이 가장 빠름을 알 수 있다. <그림 15>에서 보듯이 실험 결과를 통해 공간 미들웨어의 Intersects 연산이 Oracle Spatial과 비교해서 평균 2배의 높은 성능 향상을 보이며, ArcSDE와 비교해서 질의 영역이 약 10% 이상으로 대용량 공간 데이타를 검색할 때 공간 미들웨어의 성능이 가장 빠름을 알 수 있다.



<그림 14> Contains 연산 성능 비교

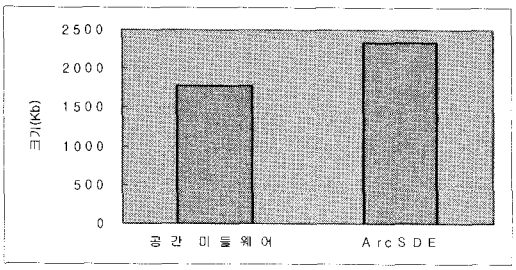


<그림 15> Intersects 연산 성능 비교

이러한 성능 평가를 통해 공간 미들웨어가 MBR 연산에 비해 공간 객체 연산에서 ArcSDE에 비해 공간 미들웨어가 대용량 공간 데이타 검색의 성능이 뛰어남을 알 수 있다. 이는 공간 미들웨어가 폴리곤의 내부를 이용한 폴리곤 내부 연산을 지원하여 공간 객체 연산의 비용을 효율적으로 줄이기 때문이다. 특히, 공간 미들웨어의 성능이 Contains 연산보다 Intersects 연산에서 좋은 이유는 Contains 연산보다 Intersects 연산에서 폴리곤 내부 연산으로 필터링되는 공간 객체의 수가 많기 때문이다.

4.6 메모리 성능

메모리 성능은 공간 미들웨어와 ArcSDE 각각에 대해서 Oracle 내부에서의 인덱스를 위한 메모리 크기를 비교하였다. Oracle Spatial의 경우 외부 프로시져 형태로 제공되기 때문에 Oracle 내부의 통계 자료를 통해서 인덱스의 크기를 확인할 수 없어서 비교 대상에서 제외하였다. <그림 16>은 각 시스템을 위한 메모리의 크기를 보여준다. <그림 16>에서 보듯이 공간 미들웨어가 ArcSDE와 비교해서 메모리의 크기가 작으며, 따라서 시스템 자원 소모가 ArcSDE에 비해 적음을 알 수 있다.



<그림 16> 메모리 크기 비교

5. 결 론

본 논문에서 개발한 대용량 공간 데이타의 효율적인 검색을 위한 공간 미들웨어는 안정적인 공간 데이타의 관리를 위해 상용 DBMS인 Oracle을 데이타 저장소로써 활용하였으며, Oracle에 저장된 대용량 공간 데이타의 효율적인 접근을 위해 OCCI를 이용하였다. 또한, 대용량 공간 데이타의 효율적 검색을 위해 다양한 공간 연산 기법 및 Array Fetch 기법 등을 사용하였다. 마지막으로, 성능 평가를 통해 공간 미들웨어의 성능을 최적으로 하기 위한 ARRAYFETCH와 타일링 레벨에 대해서 분석하였으며, 공간 미들웨어, ArcSDE, Oracle Spatial의 질의 영역에 대한 MBR 연산과 공간 객체 연산과 메모리 성능을 비교함으로써 공간 미들웨어의 대용량 공간 데이타에 대한 검색 성능과 메모리 성능이 우수함을 보였다.

참고문헌

1. 박희현, 김정준, 김동오, 홍동숙, 한기준, "ZUBICON을 이용한 웹 피쳐 서비스 시스템 설계 및 구현," GIS/RS 공동 춘계학술대회 논문집, 한국GIS학회, 2006, pp.139-145.
2. Ashdown, L., *Oracle Database Application Developer's Guide - Fundamentals 10g Release 2*, Oracle, 2005.
3. Geringer, D., *Oracle 9i: Spatial*, Oracle, 2001.
4. ESRI, *Understanding ArcSDE*, 2005.
5. Yingcheng, L., and Ling, L., "Research on Spatial Database Design and Tuning Based on Oracle and ArcSDE," Proc. of the International Congress for Photogrammetry and Remote Sensing, 2004, pp.370-375.
6. 김동오, 주성완, 홍동숙, 한기준, "대용량 위치정보·저장시스템 개발," GIS/LBS 학술대회 논문집, 6권2호, 2004, pp.105-112.
7. 신중수, 김동오, 강홍구, 박춘걸, 한기준, "Oracle 기반의 대용량 지도 서비스를 위한 XServer의 설계 및 구현," 한국공간정보시스템학회 추계학술대회 논문집, 7권3호, 2005, pp.47-54.

8. Raphaely, D., and Gregoire, J., *Oracle C++ Call Interface*, Oracle, 2001.
9. Papadias, D., Theodoridis, Y., Sellis, T., and Egenhofer, M., "Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees," Proc. of ACM SIGMOD International Conference on Management of Data, 1996, pp.92-103.
10. Kothuri, R. K., and Ravada, S., "Efficient Processing of Large Spatial Queries Using Interior Approximations," Proc. of the International Symposium on Advances in Spatial and Temporal Databases, VOL.2121, 2001, pp.404-421.
11. Ryden, K., *OpenGIS Implementation Specification for Geographic Information - Simple Feature Access - Part 1:Common Architecture*, Open Geospatial Consortium Inc., 2005.

이기영
1984년 숭실대학교 전자계산학과(공학사)
1988년 건국대학교 대학원 컴퓨터공학과(공학석사)
2005년 건국대학교 대학원 컴퓨터 · 정보통신공학과 (공학박사)
1984년 ~ 1991년 한국해양연구원 연구원
1991년 ~ 현재 을지대학교 의료산업학부 교수
관심분야 : 데이타베이스, GIS

김동오
2000년 건국대학교 컴퓨터공학과(공학사)
2002년 건국대학교 대학원 컴퓨터 · 정보통신공학과 (공학석사)
2006년 건국대학교 대학원 컴퓨터 · 정보통신공학과 (공학박사)
2006년~현재 건국대학교 컴퓨터공학부 강의교수
관심분야 : 데이타베이스, 유비쿼터스 센서 네트워크, LBS, GML, 정보시스템 감리

신중수
2005년 건국대학교 컴퓨터공학과(공학사)
2007년 건국대학교 대학원 컴퓨터 · 정보통신공학과 (공학석사)
관심분야 : 데이타베이스, GIS

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