모바일 장치를 위한 내장형 시공간 DBMS†
Embedded Spatio-Temporal DBMS for Mobile Devices

심희정*, 김정준, 신인수, 한기준
Hee-Joung Sim*, Joung-Joon Kim, In-Su Shin, and Ki-Joon Han
건국대학교 컴퓨터학과
{hjsim*, jjkim9, isshin, kjhan}@db.konkuk.ac.kr

요 약

최근 유비쿼터스 컴퓨팅 환경이 발전함에 따라 교통(u-Transport), 복지(u-Care), 문화(u-Fun), 환경(u-Green), 산업(u-Business), 행정(u-Government), 도시(u-City) 뿐만 아니라 사용자의 위치와 다양한 공간 정보를 제공하는 u-GIS가 유비쿼터스 컴퓨팅 환경의 핵심 요소 기술로 대두되고 있다. 이에 본 논문에서는 기존의 PC용 MMDBMS인 HSQLDB를 확장하여 모바일 장치에서 시공간 데이타를 효율적으로 처리 및 관리할 수 있는 내장형 시공간 DBMS를 설계 및 구현하였다. 내장형 시공간 DBMS는 OpenGIS "Simple Features Specification for SQL"에서 명시하는 공간 데이타 타입과 공간 연산자를 확장하여 시공간 데이타 타입과 시공간 연산자를 제공하며, 시공간 데이타 특성을 고려한 산술 연산 코딩 압축 기법을 제공하고, 모바일 저장 장치인 플래쉬 메모리에서 효율적인 시공간 데이타 검색을 위한 시공간 인덱스를 지원한다. 그리고, 내장형 시공간 DBMS와 u-GIS 서버 사이에서 시공간 데이타 수입/수출의 성능 향상을 위한 데이타 캐쉬 기능과 DBMS의 안정성을 위한 백업/복구 기능을 지원한다.

1. 서론

최근 유비쿼터스 컴퓨팅 환경이 발전함에 따라 교통(u-Transport), 복지(u-Care), 문화(u-Fun), 환경(u-Green), 산업(u-Business), 행정(u-Government), 도시(u-City) 뿐만 아니라 사용자의 위치와 다양한 공간 정보를 제공하는 u-GIS가 유비쿼터스 컴퓨팅 환경의 핵심 요소 기술로 대두되고 있다[1].

현재 모바일 장치는 기존 PC에 비해 작은 용량의 저장 공간과 낮은 성능의 프로세서를 사용하고 있고 자체 화일 시스템을 기반으로 데이타를 처리 및 관리하고 있다. 이는 작은 용량의 데이타를 처리 및 관리하기는 편리할지 모르나 나날이 증가하고 있는 u-GIS의 대용량의 시공간 데이타를 처리 및 관리하거나 매우 복잡하고 다양한 시공간 질의를 처리하기에는 큰 문제가 있다. 이러한 문제는 모바일 장치에서 내장형 시공간 DBMS를 이용해 시공간 데이타를 처리 및 관리함으로써 해결될 수 있다[2].

본 논문에서는 메모리 기반 관계형 DBMS인 HSQLDB[3]에 시공간 데이타 타입과

† 본 연구는 서울시 산학연 협력사업의 신기술 연구개발 지원사업의 지원으로 수행되었음.

시공간 연산자를 추가하고, 시공간 데이타 특성을 고려한 산술 연산 코딩 압축 기법, 모바일 저장 장치인 플래쉬 메모리에 최적화 하기 위해 MBR 압축 기법을 적용한 시공간 인덱스, 그리고 모바일 장치와 u-GIS 서버와의 통신 성능을 높이기 위한 데이타 캐쉬 기능등을 추가 및 확장하여 모바일 장치에서 시공간 데이타를 효과적으로 처리 및 관리할 수 있는 내장형 시공간 DBMS를 설계 및 구현하였다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 관련 연구로 내장형 시공간 DBMS의 요구사항과 HSQLDB에 대해 살펴본다. 그리고, 3장에서는 내장형 시공간 DBMS의 구조와 내장형 시공간 DBMS의 주요 기능을 설명하고, 4장에서는 내장형 시공간 DBMS의 각 모듈에 대해 상세히 설명한다. 그리고 5장에서는 다양한 성능 평가를 통해 본 논문에서 제안하는 내장형 시공간 DBMS의 우수성을 입증하고, 마지막으로 6장에서는 결론에 대하여 언급한다.

2. 관련 연구

본 장에서는 모바일 장치를 위한 내장형 시공간 DBMS의 요구 사항과 HSQLDB에 대해 설명한다.

2.1. 요구 사항

모바일 장치를 위한 내장형 시공간 DBMS는 초경량 실시간 DBMS 기술 분야에 해당하며, 모바일 장치의 단점인 느린 처리 속도와 작은 저장 공간의 한계를 극복하기 위한 기술이 필요하다. 또한, 다른 시스템에 비해 매우 큰 용량의 시공간 데이타를 처리하는 u-GIS 기능을 모바일 장치에서 수행하기 위해서는 u-GIS 서버의 대용량 시공간 데이타를 네트워크 지연 없이 모바일 장치에 전송하기 위한 데이타 압축 기능[4]과 데이타 캐쉬 기술이 필요하다. 그리고, 모바일 장치에서 고밀도로 압축된 데이타를 빠르게 검색하기 위한 모바일 시공간 인덱스 기술[5,6], 모바일 장치에서 안정적인 데이타 관리를 위한 트랜잭션 처리 기술과 백업/복구 기술 등이 요구된다. 특히, 내장형 시공간 DBMS는 시공간 데이타 지원을 위한 시공간 데이타 타입과 시공간 연산자를 지원해야 한다[7].

2.2. HSQLDB

HSQLDB는 순수 자바로 만들어진 메모리 기반 관계형 DBMS이다[3]. HSQLDB는 Hypersonic SQL 프로젝트로 시작하여 1988년 처음 배포되었다. 그리고, Hypersonic SQL의 몇몇 개발자들에 의해 HSQLDB 개발 그룹이 형성되었고, 2007년 9월에 1.8.0 버전이 배포되었다. HSQLDB의 특징은 순수 자바로 구현되었기 때문에 시스템에 독립적이며 표준 ANSI-92 SQL과 JDBC 인터페이스를 지원한다. 그리고, 빠른 검색을 위한 B+-tree를 지원하며, 데이타 무결성을 위해 트랜잭션 지원과 Outer Join, Inner Join을 지원한다. 또한, View, Trigger와 Order by, Group by, Having을 지원하며, Count, Sum, Min, Max, Avg 함수를 제공한다. 그리고, 사용자는SQL script 화일을 이용하여 데이타베이스를 생성할 수 있다.

3. 내장형 시공간 DBMS의 설계

본 장에서는 내장형 시공간 DBMS의 구조와 주요 기능에 대해 설명한다.

3.1. 내장형 시공간 DBMS의 구조

본 논문에서 개발한 내장형 시공간 DBMS는 인터페이스 관리자, 질의 처리 관리자, 인덱스 관리자, 데이타 캐쉬 관리자, 트랜잭션 관리자, 메모리 관리자, 데이타 압축 관리자, 수입/수출 관리자, 백업/복구 관

리자로 구성된다. 그림 1은 내장형 시공간 DBMS의 전체 구조를 보여준다.

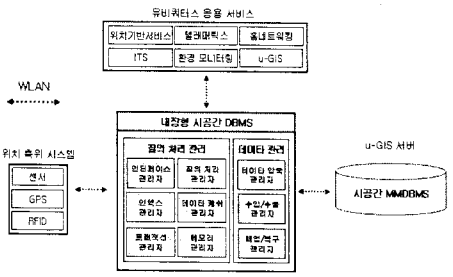


그림. 1. 내장형 시공간 DBMS 구조

수입/수출 관리자와 데이타 캐쉬 관리자는 내장형 시공간 DBMS와 u-GIS 서버 또는 유비쿼터스 응용 서비스간에 시공간 데이타 교환 기능을 제공하고, 인터페이스 관리자와 메모리 관리자는 사용자로부터 질의 입력과 모바일 장치의 메모리 관리 기능을 제공한다. 데이타 압축 관리자는 시공간 데이타 수입/수출 시 데이타 압축 기능을 제공하며, 질의 처리 관리자와 트랜잭션 관리자는 SQL 구문을 검사, 분석하여 다양한 시공간 질의를 안정적으로 처리하는 기능을 제공한다. 그리고, 백업/복구 관리자는 데이타베이스 복구 기능을 제공하고, 인덱스 관리자는 시공간 데이타 검색을 위한 시공간 인덱스를 제공한다.

3.2. 시공간 데이타 타입과 연산자

내장형 시공간 DBMS는 유비쿼터스 컴퓨팅 환경의 다양한 시공간 데이타에 대한 검색, 삽입, 삭제, 갱신 연산 수행을 위해 OpenGIS "Simple Features Specification for SQL"에서 명시하는 공간 데이타 타입과 공간 연산자를 확장하여 시공간 데이타 타입과 시공간 연산자를 제공한다 [7]. 표 1은 내장형 시공간 DBMS에서 제공하는 시공간 데이타 타입과 시공간 연산자를 보여준다.

표. 1. 시공간 데이타 타입과 시공간 연산자

시공간 데이타 타입	표 현
ST_Point	ST_POINT (2007/09/12|3:15:20 10 10)
ST_LineString	ST_LINESTRING (2007/09/12|3:15:20 10 10, 20 20, 30 40)
ST_Polygon	ST_POLYGON (2007/09/12|3:15:20 0 10, 10 20, 20 20, 20 15, 10 10, 0 10)
ST_MultiPoint	ST_MULTIPOINT (2007/09/12|3:15:20 0 10, 2007/09/12|3:15:20 0 20)
ST_MultiLineString	ST_MULTILINESTRING ((2007/09/12|3:15:20 0 10, 20 20), (2007/09/12|3:15:21 5 15, 30 15))
ST_MultiPolygon	ST_MULTIPOLYGON ((2007/09/12|3:15:20 10 10, 10 20, 20 20, 20 15, 10 10), (2007/09/12|3:15:20 10 60, 70 70, 80 60, 60 60, 10 60))
GeometryCollection	GEOMETRYCOLLECTION (ST_POINT (2007/09/12|3:15:20 10 10), ST_POINT (2007/09/12|3:15:20 30 30), ST_LINESTRING (2007/09/12|3:15:20 15 15, 20 20))

시공간 관계 연산자	설 명
ST_Equals(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B가 같은지 여부 반환
ST_Disjoint(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B가 만나지 않는지 여부 반환
ST_Touches(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B의 경계가 만나는지 여부 반환
ST_Within(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A가 객체 B에 포함되는지 여부반환
ST_Overlaps(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B가 겹치는지 여부 반환
ST_Crosses(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B가 교차하는지 여부 반환
ST_Intersects(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B가 만나는지 여부 반환
ST_Contains(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A가 객체 B를 포함하는지 여부 반환

시공간 분석 연산자	설 명
ST_Distance(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B의 거리를 반환
ST_Intersection(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B의 교집합을 반환
ST_Difference(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B의 차집합을 반환
ST_Union(ST_Geometry A, ST_Geometry B)	주 시간 단위 마다 객체 A와 객체 B의 합집합을 반환
ST_Buffer(ST_Geometry A, Double L)	주 시간 단위 마다 객체 A의 버퍼영역과 겹쳐 (L만큼 변화된 시공간 객체를 반환

시공간 검색 연산자	설 명
ST_Enters(ST_Geometry A, ST_Geometry B)	객체 A가 객체 B의 외부에서 내부로 들어가는지 여부를 반환
ST_Passes(ST_Geometry A, ST_Geometry B)	객체 A가 객체 B의 외부에서 내부로 들어왔다가 다시 외부로 나가는지 여부를 반환
ST_Meets(ST_Geometry A, ST_Geometry B)	객체 A가 객체 B의 외부에서 경계만 터치하고 지나가는지 여부를 반환
ST_Leaves(ST_Geometry A, ST_Geometry B)	객체 A가 객체 B의 내부에서 외부로 나가는지 여부를 반환
ST_Inside(ST_Geometry A, ST_Geometry B)	객체 A가 객체 B의 내부에만 존재하는지 여부 반환

3.3. 시공간 데이타 압축

본 논문에서는 시공간 데이타 압축 방법으로 기준점과의 거리차를 이용한 산술 연산 코딩 압축 기법을 적용하였다. 일반적으로 GIS에서 가장 많은 크기를 차지하는 ST_LineString이나 ST_Polygon은 특정 부위에 대하여 클러스터링된 형태를 띠게 된다. 그러므로, 각 객체의 첫 번째 좌표를 기준으로 하여 거리의 차이를 구하게 되면 기준점의 데이타는 8바이트를 차지하게 되지만 그 이후 정수측의 차이 값이 작아지게 된다. 즉, double 형으로 하나의 점을 표시할 때 16바이트가 소모되는 것을 최소 4바이트, 최대 12바이트로 표현할 수 있게 된다.

그림 2는 산술 연산 코딩 압축 구조와 거리 차이가 +7.878100일 경우 산술 연산 코딩 압축 기법으로 표현한 예를 보여준다.

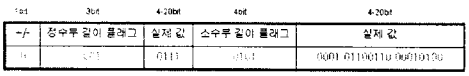


그림. 2. 산술 연산 코딩 압축 구조

본 논문에서는 시공간 데이터의 압축률을 더욱 높이기 위해 산술 연산 코딩 압축 기법을 개선하였다. 이전의 산술 연산 코딩 압축 기법은 각 객체의 첫 번째 좌표점들은 크기를 줄일 수 없는 단점을 가지고 있다. 따라서, 본 논문에서는 두 번째 객체들부터 각 객체의 첫 번째 좌표점을 이전 객체의 첫 번째 좌표점과의 거리차를 구함으로써 각 객체들의 첫 번째 좌표점들도 정수측 값을 줄여 압축 성능을 더욱 향상시켰다. 이러한 방법은 객체들이 밀집되어 있을 경우 더욱 높은 압축률을 가질 수 있게 된다. 그림 3은 본 논문에서 제안하는 산술 연산 코딩 압축 기법을 이용한 시공간 데이타 압축의 예를 보여준다.

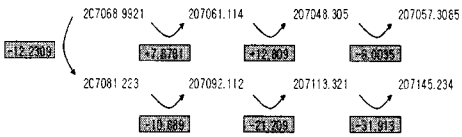
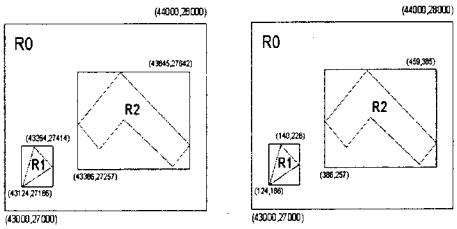


그림. 3. 시공간 데이타 압축 예

첫 번째 ST_Polygon의 X좌표 207068.9921과 두 번째 ST_Polygon의 X좌표 207081.223의 차이 -12.2309를 본 논문에서 제시한 압축 구조로 표현하게 되면 기존의 산술 연산 코딩 압축 기법에서 각 ST_Polygon의 첫 번째 좌표들의 값은 줄일 수 없었던 단점을 해결하여 더욱 압축 성능을 높일 수가 있게 된다.

3.4. 시공간 인덱스

내장형 시공간 DBMS는 시공간 데이타의 효율적인 검색을 위하여 다차원 인덱스를 지원한다. 본 논문에서는 이러한 다차원 인덱스를 모바일 저장 장치인 플래쉬 메모리에 최적화 시키기 위하여 엔트리 크기를 줄일 수 있는 RSMBR(Relative-Sized MBR) 압축 기법을 제시하였다. RSMBR 압축 기법은 인덱스 생성 및 갱신 비용을 줄이기 위하여 상대 좌표와 크기를 이용하여 MBR을 압축한다. 즉, MBR 압축 시 MBR이 커지는 현상을 제거하기 위하여 MBR의 좌하점은 상대 좌표값으로 표현하고 우상점은 MBR의 크기로 표현한다. 그림 4는 본 논문에서 제시하는 RSMBR 압축 기법을 적용한 예를 보여준다.



(a) 절대 좌표 MBR **(b) RSMBR**

그림 4.MBR 압축 기법

그림 4(a)는 절대 좌표 MBR로 표현된 그림이고, 그림 4(b)는 본 논문에서 제안하는 RSMBR 압축 기법으로 표현한 그림이다. 그림 4(b)에서 객체 R1의 RSMBR은 (124, 186), (140, 228) 이고, 객체 R2의 RSMBR은 (386, 257), (459, 385)이다. RSMBR 압축 기법에서 각 우상점 좌표값은 길이 플래그와 실제 값을 이용하여 저장된다. 그림 5는 RSMBR 좌표값의 데이타 저장 구조를 보여준다.

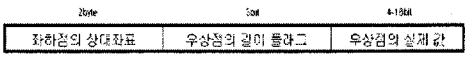


그림 5. RSMBR 좌표값의 데이타 저장 구조

그림 4(b)에서 객체 R1의 RSMBR 우상점 좌표값 중 140은 길이 플래그 010과 실제 값 10001100로 표현될 수 있고, 객체

R2의 RSMBR 우상점 좌표값 중 459는 길이 플래그 011과 실제 값 0001110010 11로 표현될 수 있다. 결론적으로 RSMBR의 각 X, Y 좌표값은 최소 6byte, 최대 10byte로 표현 가능하기 때문에 전체적인 MBR의 저장 공간이 줄어들게 된다.

RSMBR 압축 기법은 RSMBR 재구성을 위한 추가 갱신 비용을 줄이기 위하여 넓은 분포의 경우와 좁은 분포의 경우로 나누어 압축 기준점을 다르게 적용한다. 넓은 분포란 전체 MBR의 크기가 일정 크기보다 큰 경우이고 좁은 분포란 전체 MBR 크기가 일정 크기보다 같거나 작은 경우를 말한다. 본 논문에서는 넓은 분포와 좁은 분포를 구분하기 위한 기준 크기를 4byte로 하였다. 그림 6과 그림 7은 넓은 분포일 경우와 좁은 분포일 경우의 RSMBR 재구성 방법을 보여준다.

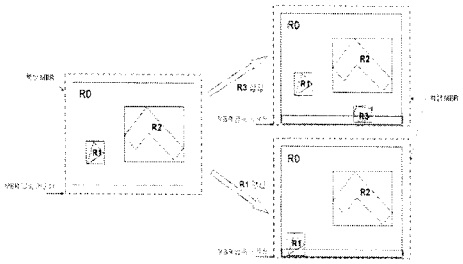


그림 6. 넓은 분포일 경우 RSMBR 재구성

그림 6에서와 같이 객체 R3이 삽입될 경우 부모 노드 R0의 MBR이 변경되더라도 R3의 MBR이 R1, R2, R3의 부모 노드인 R0의 확장 MBR에 포함되기 때문에 RSMBR 재구성이 필요 없다. 또한, 객체 R1이 갱신될 경우에도 R1의 MBR이 R0의 확장 MBR에 포함되기 때문에 RSMBR 재구성이 필요 없다. 이렇게 넓은 분포일 경우 부모 노드 확장 MBR의 좌하점을 기준으로 압축하면 부모 노드의 MBR 변화에 둔감하기 때문에 RSMBR 재구성을 위한 추가 갱신 비용을 줄일 수 있게 된다.

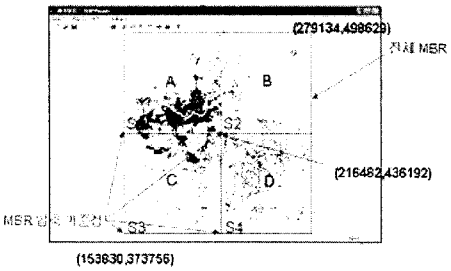


그림 7.좁은 분포일 경우 RSMBR 재구성

그림 7에서 전체 MBR의 크기가 일정 크기(4byte) 보다 같거나 작기 때문에 전체 MBR을 2byte의 셀(A, B, C, D)로 나누어 RSMBR 압축 기준점을 적용하였다. 즉, A 셀에 포함되는 객체들은 A 셀의 좌하점인 S1을 기준으로 압축되고 B 셀에 포함되는 객체들은 B 셀의 좌하점인 S2를 기준으로 압축된다. 그리고 C 셀과 D 셀에 포함되는 객체들도 각각 C 셀과 D 셀의 좌하점인 S3과 S4를 기준으로 압축된다. 이렇게 좁은 분포일 경우 전체 MBR을 2byte 단위의 셀로 나누고 각 셀의 좌하점을 RSMBR 압축 기준점으로 적용하면 RSMBR 재구성을 위한 추가 갱신 비용을 줄일 수 있게 된다.

4. 내장형 시공간 DBMS의 구현

본 장에서는 내장형 시공간 DBMS의 각 관리자에 대해 상세히 설명한다.

4.1. 질의 처리와 인덱스 그리고 인터페이스 관리자

질의 처리 관리자는 시공간 데이타 타입과 시공간 연산자를 이용하여 유비쿼터스 컴퓨팅 환경의 다양한 시공간 데이타에 대한 검색, 삽입, 삭제, 갱신 연산을 수행한다.

인덱스 관리자는 시공간 데이터의 효율적

인 검색을 위해 시공간 인덱스를 제공한다. 이러한 시공간 인덱스를 모바일 저장 장치인 플래쉬 메모리에 최적화하기 위해 RSMBR(Relative-Sized MBR) 압축 기법을 사용한다.

인터페이스 관리자는 사용자로부터 질의 입력과 질의 결과를 모바일 장치의 화면에 지도로 디스플레이하는 기능을 제공한다.

4.2. 수입/수출과 데이타 캐쉬 관리자

수입/수출 관리자는 유비쿼터스 응용 서비스와 u-GIS 서버 등 다양한 시스템과의 데이타 교환을 위해 표준 인터페이스(JDBC)를 제공한다.

데이타 캐쉬 관리자는 모바일 장치에서 시공간 데이타의 검색 효율을 높이기 위해 사용자의 위치를 기반으로 요청이 예상되는 주변 데이타를 관리하는 기능과 검색된 데이타를 중요도에 따라 레이어로 관리하는 기능을 제공한다.

4.3. 메모리와 데이타 압축 관리자

메모리 관리자는 대용량의 시공간 데이타를 모바일 장치의 메인 메모리에서 효율적으로 관리하기 위하여 질의 처리 시 필요한 메인 메모리 할당과 반납에 따른 성능 저하를 최소화한다.

데이타 압축 관리자는 내장형 시공간 DBMS와 u-GIS 서버 또는 유비쿼터스 응용 서비스 간에 시공간 데이타 교환시 네트워크 효율성을 높이기 위해 산술 연산 코딩 압축 기법을 이용한 데이타 압축 기능을 제공한다.

4.4. 백업/복구와 트랜잭션 관리자

백업/복구 관리자는 모바일 시공간 DBMS 메모리상의 데이타를 일정 주기마다 화일로 백업하여 데이타의 안정성과 시스템의 고가용성을 제공하고 시스템의 로그 화일을 이용하여 커밋된 트랜잭션, 진행 중인 트랜잭션, 취소된 트랜잭션을 정확히 구분하고 회복시점에 반영함으로써 장애 이전의 상태로 데이타베이스를 완벽히 복구할 수 있는 기능을 제공한다.

트랜잭션 관리자는 모바일 장치의 메인 메모리 특성에 맞게 최적화된 알고리즘을 이용하여 질의 처리 시 고성능의 트랜잭션 관리 기능을 제공하고 커밋(Commit)과 롤백(Rollback) 처리를 지원하여 데이타 무결성을 보장한다.

5. 성능 평가 및 분석

본 장에서는 내장형 시공간 DBMS의 성능 평가 및 분석에 대하여 설명한다.

5.1. 성능 평가 환경 및 테스트 데이타

내장형 시공간 DBMS는 모바일 장치로 Compaq iPAQ hx2190와 NAND 플래쉬 메모리(4GB)를 u-GIS 서버로 실시간 시공간 MMDBMS를 사용하였다. 구현은 Windows Mobile 5.0에서 Personal Java 1.1를 사용하였고 컴파일 툴로 ANT를 사용하였다.

테스트 데이타는 이동체 데이타 생성기인 GSTD(Generate Spatio Temporal Data) [8]와 GSTMO(Generating Semantics-Based Trajectories of Moving Object) [9]를 사용하여 생성하였다.

5.2. 데이타 압축 성능 평가

데이타 압축 성능 평가는 LZO 압축 기법 [10]과 본 논문에서 제안한 산술 연산 코딩 압축 기법을 비교 분석하였다. 그림 8과 그림 9는 산술 연산 코딩 압축 기법의 압축률과 압축 시간에 대한 성능 평가 결과를 보여준다.

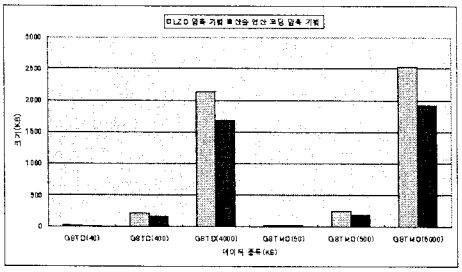


그림 8. 데이타 압축률

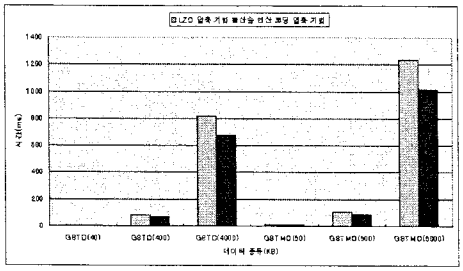


그림 9.데이타 압축 시간

그림 8과 9에서 보여주는 바와 같이 산술 연산 코딩 압축 기법이 LZO 압축에 비해 압축률은 20~25%, 압축 시간은 15~20% 향상된 결과를 보여주었다. 이것은 일반적인 데이타가 아닌 시공간 데이타 압축에서는 LZO 압축보다 산술 연산 코딩 압축 기법이 시공간 데이타 특성을 고려 하였기 때문에 압축률이 더욱 좋아지고 압축 방식이 간단하기 때문에 압축 시간도 줄어 들게 된다는 것을 의미한다.

5.3. 인덱스 성능 평가

인덱스 성능 평가는 절대 좌표 MBR을 사용하는 3DR-tree[11]와 본 논문에서 제안한 RSMBR을 적용한 RS-3DR-tree(RSMBR based 3DR-tree)를 비교 분석하였다. 그림 10은 갱신 대상 객체의 수가 각각 전체의10%, 20%, 30%, 40%, 50%일 때 3DR-tree와 RS-3DR-tree에 대해서 데이타 갱신 시간을 비교한 것을 보여주고 그림 11은 질의 영역이 각각 전체의 10%, 20%, 30%, 40%, 50%일 때 3DR-tree와 RS-3DR-tree에 대해서 데이타 검색 시간을 비교한 것을 보여준다.

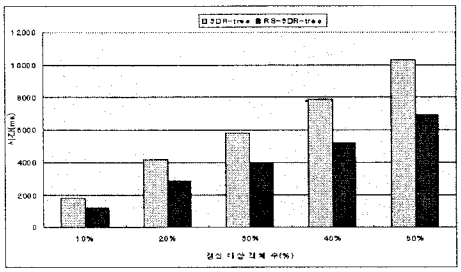


그림 10. 데이타 갱신 시간

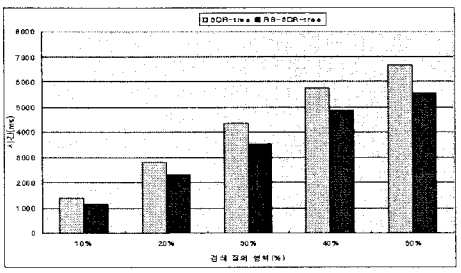


그림 11. 데이타 검색 시간

그림 10에서 보여주는 바와 같이 각각의 인덱스에서 데이타 갱신 시간은 RS-3DR-tree가 3DR-tree 보다 30~35% 성능이 향상된 것을 알 수 있다. 이것은 RS-3DR-tree가 RSMBR 압축 기법을 적용하였기 때문에 모바일 저장 장치인 플래쉬 메모리 연산에서 가장 많은 비용을 차지하는 쓰기 연산을 3DR-tree보다 더 적게 수행하면서 데이타 갱신을 수행한다는 것을 의미한다.

그림11에서 보여주는 바와 같이 각각의 인덱스에서 데이타 검색 시간은 RS-3DR-tree가 3DR-tree 보다 15~20% 성능이 향상된 것을 알 수 있다. 이것은 RS-3DR-tree가 RSMBR 압축 기법 적용하였기 때문에 정해진 크기의 노드 안에 더 많은

엔트리를 저장할 수 있으므로 모바일 저장 장치인 플래쉬 메모리 연산에서 3DR-tree보다 읽기 연산을 더 적게 수행하면서 데이타 검색을 수행한다는 것을 의미한다.

6. 결론

유비쿼터스 컴퓨팅 환경에서 대용량의 시공간 데이타를 신속하게 처리하고 효과적으로 관리하기 위해서는 모바일 장치에서 시공간 데이타를 효율적으로 처리 및 관리할 수 있는 내장형 시공간 DBMS가 필요하다.

본 논문에서 설계 및 구현한 내장형 시공간 DBMS는 유비쿼터스 환경에서 필요한 시공간 데이타를 효율적으로 관리하기 위해 시공간 데이타 타입과 시공간 연산자를 지원하고, 내장형 시공간 DBMS와 u-GIS 서버간에 시공간 데이타 교환 시 네트워크 효율성을 위해 시공간 데이타 특성을 고려한 산술 연산 코딩 압축 기능을 제공한다. 또한, 시공간 데이타의 빠른 검색을 위해 모바일 저장 장치인 플래쉬 메모리에 최적화된 시공간 인덱스를 제공하고 모바일 장치에서 u-GIS 서버와의 통신 성능을 높이기 위한 데이타 캐쉬 기능, 데이타의 안정성과 시스템의 고가용성을 위한 백업/복구 기능을 제공한다.

참고문헌

1. Kim, J. J., Hong, D. S., Kang, H. K., and Han, K. J., "TMOM: A Moving Object Main Memory-Based DBMS for Telematics Services," Proc. of the W2GIS'06, 2006, pp. 259-268.

2. Chang, L. P., and Kuo, T. W., "An efficient Management Scheme for Large Scale Flash-Memory Storage Systems," Proc. of the ACM SAC'04, 2004, pp. 862-868.

3. HSQLDB: http://hsqldb.sourceforge.net, 2007.

4. Kim, K. H., Cha, S. K., and Kwon, K. J., "Optimizing Multidimensional Index Tree for Main Memory Access," Proc. of the ACM SIGMOD Record, 2001, pp. 139-150.

5. Byun, S. W., Huh, M. H., and Hwang, H. Y., "An index rewriting scheme using compression for flash memory database systems," Journal of Information Science, 2007, pp. 398-415.

6. Wu, C. H., Chang, L. P., and Kuo, T. W., "An Efficient R-Tree Implementation over Flash-Memory Storage Systems," Proc. of the ACM GIS'03, 2003, pp. 17-24.

7. OpenGIS Consortium, Inc., Simple Features Specification For SQL 1.1, 1999.

8. Theodoridis, Y., Silva, J. R. O., and Nascimento,M. A., "On the Generation of Spatiotemporal Datasets," Proc. of the SSD'99, 1999, pp. 147-164.

9. Pfoser, D., and Thedoridis, Y., "Generating Semantics-Based Trajectories of Moving Object," Proc. of the International Workshop on Emerging Technologies for Geo-Based Applications, 2000, pp. 59-76.

10. LZO: http://www.oberhumer.com/opensource/lzo, 2005.

11. Theodoridis, Y., Vazirgiannis, M., Sellis, T. K., "Spatio-temporal indexing for large multimedia applications," Proc. of the 3rd IEEE International Conference on Multimedia Computing and Systems, 1996, pp. 441-448.