DOI: 10.3745/KIPSTD.2008.15-D.5.609

부분 맵 업데이트 지원 내비게이션을 위한 모바일 공간 DBMS 개발 및 성능 평가

민 경 욱[†] · 안 경 환[††] · 김 주 완[†††] · 진 성 일[††††]

요 약

모바일 단말에서의 맵 데이터 활용 서비스는 내비게이션, 위치기반서비스 등 그 활용 범위가 다양해지고 있다. 이러한 모바일 단말에서 저장/관리되는 맵 데이터의 크기는 갈수록 커져 수 GB에 이른다. 기존 내비게이션 시스템의 경우, 리소스 제약으로 인해 모바일 단말의 성능을 최대로 활용하기 위하여 맵 데이터를 읽기 최적화된 물리적인 저장형식(PSF: Physical Storage Format)으로 저장 관리하고 있다. 즉, 맵 데이터가 변경된 경우 변경된 부분만의 업데이트가 불가능하여 전체 데이터를 업데이트해야 하는 단점이 있다. 일반적으로 2 GB 데이터를 모바일 단말의 플래시메모리에 기록하는 시간은 수십 분이 소요된다. 따라서 본 연구에서는 이러한 기존의 내비게이션용 맵 데이터의 부분 업데이트가 불가능한 점을 해결하고자 모바일 공간 DBMS을 개발하였다. 그리고, 모바일 공간 DBMS에서의 내비게이션 서비스의 성능을 보장하기 위한 기법을 제시하고 실험을 통하여 이를 검증하였다.

키워드 : 모바일 DBMS, 임베디드 DBMS, 내비게이션, GIS, 텔레매틱스, 부분 맵 업데이트

The Development and Performance Evaluation of the Mobile Spatial DBMS for the Partial Map Air Update in the Navigation

Min Kyoung Wook[†] · An Kyoung Hwan[††] · Kim Ju Wan[†††] · Jin Sung Il[††††]

ABSTRACT

The service handling the map data in the mobile device including navigation, LBS, Telematics, and etc., becomes various. The size of map data which is stored and managed in the mobile device is growing and reaches in several GB. The conventional navigation system has used the read-only PSF (physical storage format) in order to enhance the performance of system by maximum in the mobile device which has limited resources. So though a little part of the map data is changed the whole data must be updated. In general, it takes several ten minutes to write the 2 GB map data to a flash memory of mobile device. Therefore, we have developed the mobile spatial DBMS (database management system) to solve the problem which is that the partial map data couldn't be updated in the conventional navigation system. And we suggest the policy to guarantee the performance of the navigation system which is implemented using the spatial mobile DBMS and verify this by experiment.

Keywords : Mobile DBMS, Embedded DBMS, Navigation, GIS, Telematics, Map Air Update

1. 서 론

모바일 단말에서는 계산능력과 자원이 풍부해짐에 따라 대용량의 데이터를 관리할 수가 있다. 가장 대표적인 서비스는 내비게이션 서비스이며, 최근에는 약 4Gbyte에 이르는 데이터를 이용하는 서비스가 출시되고 있다. 이 중에서 가장 큰 부분을 차지하는 데이터는 맵 데이터이다. 이러한 맵 데이터는 지도 화면 표출, POI(Point Of Interest) 검색, 경로 검색에 활용이 되며 얼마나 상세한 맵 데이터를 제공하느냐에 따라서 서비스의 품질이 결정된다.

기존 내비게이션 서비스에서의 맵 데이터 변경은 수개월에 한번씩 전체 맵 데이터를 변경하도록 서비스를 제공한다. 이 업데이트 서비스는 플래시 메모리에 전체 데이터를 재 기록(rewrite)하도록 하고 있으며 약 2 GB의 전체 데이터를 기록하는 데에는 수십 분이 소요된다. 실제 변경된 데이터가 일부분임에도 불구하고 전체 데이터를 변경해야 한다. 지금까지 모바일 단말에서 저장/관리되는 맵 데이터에 대해서 부분 업데이트를 하지 못하는 이유는 내비게이션 성능을 최대로 보장하기 위하여, 맵 데이터 파일이 단말에 최적화 된

† 정 회 원 : ETRI 텔레매틱스연구부 선임연구원
†† 정 회 원 : ETRI 텔레매틱스연구부 선임연구원
††† 정 회 원 : ETRI 텔레매틱스연구부 팀장
†††† 종신회원 : 충남대학교 전자정보통신공학부 교수

논문접수 : 2008년 5월 30일
수 정 일 : 1차 2008년 7월 18일
심사완료 : 2008년 7월 21일

PSF 형태로 저장되어 있기 때문이다. 더군다나, 내비게이션에서 가장 중요한 도로 네트워크 데이터의 경우, 노드와 링크가 복잡한 관계를 가지고 있어 하나의 링크 또는 노드가 변경될 경우 다수 데이터가 변경되어야 하며 이럴 경우 최적화 되어 있는 파일의 구조를 유지할 수가 없다. 즉, 지금까지 모바일 단말에 저장되어 있는 맵 데이터는 부분 업데이트가 용이하지 않은 읽기 전용의 포맷을 유지하여 왔다.

세계적으로도 이러한 문제점을 해결하고자 모바일 DBMS을 이용하려는 시도가 보이고 있다. DBMS에서는 데이터 레코드의 삽입/삭제/변경은 기본적인 기능이며, 데이터의 효율적인 관리 측면에서도 그 효과가 크기 때문이다. 하지만, 이런 상용 모바일 DBMS의 경우 공간 데이터 타입, 공간 검색을 위한 색인 및 질의처리 인터페이스를 제공하고 있지 않기 때문에 내비게이션 응용에서 맵 데이터 화면 표출을 위한 공간 검색의 성능을 보장할 수가 없다. 또한 가장 중요한 경로검색의 경우에도 그 성능을 보장할 수가 없다. 그 이유는, 네트워크 데이터에 경로검색을 위한 노드/링크의 추출을 각각의 테이블을 query-by-query를 반복 수행하여 검색 할 경우 그 성능이 저하될 수밖에 없다.

이에 본 연구에서는 기존 모바일 단말에서 저장/관리되고 있는 맵 데이터의 부분 업데이트가 불가능한 문제점을 해결하기 위한 접근 방법으로 모바일 공간 DBMS를 개발하고 이를 이용하여 내비게이션을 구현하는 것에 초점을 맞춘다. 이를 위한 요구사항으로 다음 두 가지를 충족하여야 한다.

① 부분 업데이트가 가능한 데이터는 일반 속성 데이터뿐만 아니라 배경 표출용 공간 데이터, 검색용 POI 공간 데이터 및 경로 계산을 위한 도로 네트워크 데이터에 해당한다. 이러한 데이터의 부분 업데이트는 유선 인터넷뿐만 아니라, 무선 통신 서비스(CDMA, Wibro 등)를 이용하여 업데이트가 가능하여야 한다.

② 부분 업데이트가 가능한 ①의 시스템을 이용하여 구현한 내비게이션 응용에서는 화면 표출, POI 검색 및 경로 검색의 성능을 보장하여야 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 관련 연구에 대해서 살펴보고, 3장에서는 본 연구에서 제안하는 모바일 공간 DBMS에 대해서 자세히 기술한다. 그리고 4장에서는 3장의 모바일 공간 DBMS를 이용하여 구현한 내비게이션 응용 및 실제 무선 통신을 이용한 맵 업데이트(이하 맵 Air 업데이트)를 위한 시스템 구성에 대해서 살펴보고 5장에서는 실험을 통하여 모바일 DBMS와 내비게이션 응용 시스템의 성능을 분석한다. 마지막 6장에서는 결론을 맺는다.

2. 관련 연구

2 장에서는 본 연구와 관련된 관련연구에 대해서 살펴 본다. 관련연구는 크게 모바일 DBMS와 내비게이션 데이터의 부분 업데이트 기술로 분류하여 기술하고자 한다.

2.1 모바일, Embedded DBMS 관련 연구

현재 모바일 DBMS는 다양한 산업에 적용되어 활용되고 있다. 기존에 연구되었던 모바일 데이터베이스의 이슈는 데이터 동기화, 모바일 트랜잭션 처리, Small footprint, 플래시메모리 특성 등이 있다. [1]에서는 서버와 모바일 단말간의 데이터베이스 동기화와 모바일 트랜잭션 처리 및 임베디드 데이터베이스에 대한 필요 기술에 대해서 기술하고 있다. 모바일 단말의 경우 항상 서버와 연결되어 있지 않기 때문에, 서버와 데이터 중복이 발생하고 서버와의 연결에 의해서 자동적으로 데이터의 동기화가 이루어 져야 한다. 그리고 모바일 트랜잭션 처리의 경우, 일반적으로 모바일 DBMS는 다중 사용자를 지원하지 않지만, 회복 기능을 지원하여 데이터간의 일관성을 유지하여야 한다. 모바일 단말은 리소스가 제한적이며 특히 모바일 데이터베이스에서의 질의 처리는 unbound된 메모리를 이용하여 효율적인 처리로 인한 질의 처리 성능을 보장하여야 한다[4].

[2]에서는 모바일 DBMS의 small footprint에 대해서 강조하고 있으며, 일반적으로 모바일 단말의 응용은 모바일 DBMS 라이브러리를 포함하여 동작하기 때문에 그 코드 사이즈가 최소화가 되어야 한다. 모바일 DBMS의 코드 사이즈의 최소화는 서버 DBMS의 전체 기능을 포함시키지 않고 오버헤드를 최소화 하기 위해 레코드, 테이블, 컬럼 등과 같은 데이터 구조의 수와 크기를 제한한다. 또한 DBMS 기능들을 컴포넌트화하여 필요한 기능만을 이용할 수 있도록 개발될 필요가 있다. [6]의 SQL Server for Windows CE, [7]의 Oracle 9i Lite, [8]의 Sybase Adaptive Server Anywhere, [9]의 IMB DB2 Everyplace 등의 제품들은 이러한 모바일 단말의 리소스 제약에 적응적인 DBMS small footprint을 구현한 것들이다.

[3, 5]에서는 모바일 DBMS의 플래시 메모리 저장 매체의 특징에 대해서 기술하였다. 플래시 메모리의 저장 용량이 대용량화 되어 가고 있으며, 저장 매체의 성능과 수명 향상을 위한 주요 이슈들을 기술하고 있다. 플래시 메모리는 일반 하드 디스크의 저장 시스템과 다른 접근 API를 지원한다. 즉, 일반적인 저장시스템 라이브러리에서 읽기/쓰기 API를 제공하는 대신 플래시메모리에서는 읽기/쓰기/삭제 API를 제공하며 이 차이를 FTL(Flash Translation Layer)이 중간에서 맞추는 역할을 한다. 또한 플래시 메모리의 가장 큰 특징은 읽기/쓰기/삭제 비용이 큰 차이가 있으며, 처리 단위가 서로 다르다는 점이다.읽기/쓰기의 경우 일반적으로 섹터(sector or page - 512 bytes) 단위로 처리되며, 삭제의 단위는 블록(block - 16Kb)이다. 플래시메모리의 경우 이미 기록되어 있는 섹터에 재 기록을 할 경우 삭제 연산을 먼저 수행해야 하기 때문에 성능이 많이 떨어진다. 따라서, FTL에서는 이러한 in-place update 대신 out-place update를 수행하도록 동작한다.

[10]에서는 모바일 단말의 메인 메모리를 저장매체로 활용하는 Spatial MMDBMS에 대해서 개발하였다. 기존의 관계형 DBMS에 공간 데이터 타입 및 색인 등을 지원하고 공간 데이터 질의처리가 가능하도록 시스템을 확장하여 개발

하였다. 본 연구와 공간데이터 질의처리를 지원하는 점은 동일하지만, 본 연구에서는 대용량 데이터를 처리할 수 있도록 하기 위하여 플래시메모리를 저장 매체로 활용한다는 점에서 차이가 있다.

이 이외에도 모바일 데이터베이스에서는 사용자의 모니터링 없이 시스템이 수행 될 수 있는 신뢰성, zero configuration, Embedded DBMS를 다양한 플랫폼에 포팅이 가능한 Portability 등의 다양한 요구사항을 만족하여야 한다.

2.2 내비게이션 용 부분 맵 업데이트 관련 연구

기존 상용화되어 있는 내비게이션의 맵 데이터는 PSF 구조를 따른다. PSF 구조는 [11]의 KIWI 포맷, [12]의 KIWI+ 포맷, [13]의 GDF 포맷, [14]의 NAVIKEN 포맷 등이 있으며 모바일 단말의 저장 매체에 최적화 되어 있는 형태로 구성 되어 있다. 그 이유는 적은 용량의 압축과 맵 데이터의 시각화 및 경로 계산의 성능을 보장하기 위해서이다. 하지만, 이러한 PSF의 구조는 파일 시스템을 기반으로 복잡한 구조(물리적인 Offset 연결)로 되어 있으며, 데이터 생성과 관리가 어렵고 상호 호환성이 부족한 단점이 있다. 또한 맵의 부분 업데이트가 불가능하여 전체 맵을 업데이트 함으로써 많은 시간과 비용이 낭비된다.

[15]의 ActMap 프로젝트에서는 2004년부터 부분 맵 업데이트를 위한 교환 포맷 및 서버, 단말 시스템에 대한 연구를 진행하여 프로토타입 수준의 시스템을 만들었으나 상용화에 실패하였다.

[16]에서는 PSF 구조의 단점을 극복하고자 Embedded Spatial DBMS 프로토타입을 구현하였다. 오픈 소스인 SQL Lite를 이용하여 공간데이터 질의 처리기능 및 내비게이션 기능을 추가하였다. 좀 더 상세히 기술하면, 공간 데이터 인덱스와 공간 연산자, 네트워크 연산자 등을 SQL Lite 엔진 라이브러리를 이용하여 구현하였다. 초점을 둔 기능은 공간데이터 압축과 계층적 도로 네트워크 구조 지원을 위한 Multi-Link 기법, Grid 공간 인덱스 구현 등이다. 논문에서는 네트워크의 계층적 구조 지원을 위하여 Multi-Link 기법을 활용하고 있지만, 네트워크의 부분 업데이트의 로직이 복잡해지며, 업데이트 데이터양도 증가한다. 그리고 일반적으로 내비게이션용 맵 데이터는 메쉬(Mesh)에 의해서 구분되어 관리되며, 각각의 메쉬에 포함되어 있는 노드/링크 데이터에 대해서 테이블을 생성할 경우 경로검색을 위하여 다수 테이블을 검색해야 하기 때문에 성능이 저하될 수 있다.

내비게이션용 부분 맵 업데이트를 지원하는 상용화 된 시스템 중, [17]의 혼다 인터나비 내비게이션에서는 주요 도로 네트워크 데이터에 대해서만 실시간 추가를 지원하지만 변경 및 삭제는 지원하지 않는다.

이에 본 연구에서는 모바일 단말에서 동작하는 공간데이터 지원 모바일 DBMS와 이를 이용한 Air 맵 업데이트 및 내비게이션 구현에 관한 시스템을 개발하고 성능을 평가하였다.

3. 모바일 공간 DBMS

본 연구에서 개발한 모바일 공간 DBMS와 내비게이션 및 동기화 컴포넌트 소프트웨어 엔진을 FUNs (Flash-aware Ubiquitous Navigation System)라 명명한다. 3장에서는 본 연구에서 설계 및 구현한 모바일 공간 DBMS의 개념과 주요 기능 등에 대해서 자세히 살펴보고자 한다.

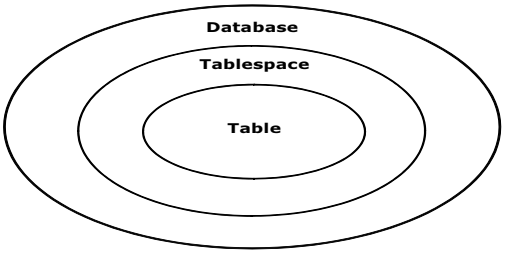
3.1 모바일 공간 DBMS의 구조

FUNs의 모바일 공간 DBMS의 개념적인 모델은 (그림 1)과 같다.

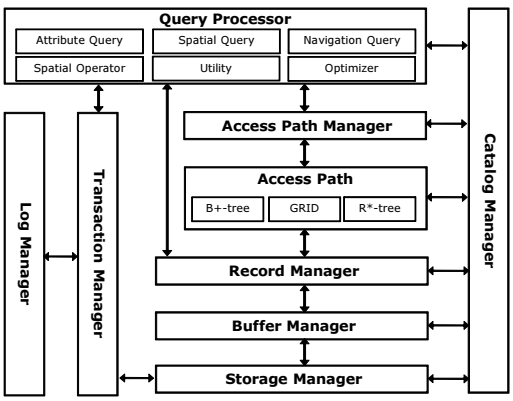
개념적인 저장 구조는 데이터베이스, 테이블스페이스 및 테이블로 구성된다. 데이터베이스는 다수 테이블스페이스로 구성된다. 테이블스페이스는 같은 유형의 데이터 군을 테이블 단위로 저장하며 테이블스페이스당 하나의 파일이 생성된다. 예를 들면, 내비게이션의 경우 화면 표출과 검색용 데이터를 하나의 테이블스페이스(Map.tbs)에 할당하고 경로검색을 위한 도로 네트워크 데이터에 해당하는 NODE, LINK 등의 테이블을 하나의 테이블스페이스(Network.tbs)에 할당할 수 있다.

FUNs의 모바일 공간 DBMS의 시스템 내부 구조는 (그림 2)와 같다. FUNs의 DBMS 구조는 일반 DBMS의 유사한 구조를 가지며 각 모듈에 대한 설명은 다음과 같다.

Storage Manager: Storage Manager는 가장 하위에 존재하며, 데이터베이스, 테이블스페이스, 테이블 생성/삭제 및 페이지 단위의 저장소를 관리한다. DBMS에서의 데이터 접근 단위는 페이지 단위이며 상위 버퍼 관리자에 의해서 데이터 페이지의 Allocation/DeAllocation/Read/Write 요청을 처리한다. 그리고 데이터 페이지의 할당과 할당해제의



(그림 1) FUNs 모바일 공간 DBMS 개념 모델



(그림 2) FUNs 모바일 공간 DBMS 구조

관리를 위하여 별도 디렉토리 페이지 관리 등의 역할을 수행한다. 본 연구에서의 Storage Manager에서는 페이지 할당과 해제 등의 기능을 OS에서 제공하는 파일 시스템과 플래시메모리 드라이버 등을 직접적으로 제어하도록 구현하니 않았으며, 이 부분에 대해서는 향 후 연구과제로 남겨둔다.

Buffer Manager: 버퍼 관리자는 저장소의 입출력 속도를 향상시키기 위해서 메인 메모리상에서 페이지 단위로 데이터를 캐쉬하는 모듈이다. 메모리 할당 용량은 전체 데이터 셋에 비하여 작으므로, LRU(Least Recently Used) 교체 전략을 이용하여 최근에 참조된 페이지가 메모리에 오래 머무를 수 있도록 해준다. 상위의 레코드 관리자는 저장 관리자에 직접 페이지 읽기/쓰기 요청을 할 수 있으나 버퍼 관리자에 페이지 입/출력을 요청함으로써 성능을 향상시킬 수 있다. 플래시 메모리의 경우 페이지가 업데이트 되는 경우 FTL에서는 해당 페이지를 재기록을 하지 않고 무효화 시킨 후 다른 블록의 페이지에 변경된 페이지를 기록한다. 만약 빈번한 페이지의 변경은 플래시메모리의 이러한 로직으로 인해 성능을 저하 시킬 수 있다. 따라서 본 모바일 DBMS의 버퍼관리자에서는 변경된 페이지의 경우 교체 후보(Replacement Victim)로 선택되지 않고, 가장 늦게 교체하도록 하는 전략을 채택하여 구현하였다.

Record Manager: 레코드 관리자는 페이지 내에서 레코드를 관리하는 역할을 수행한다. 레코드의 종류는 고정길이 레코드 페이지, 가변길이 레코드 페이지, short 가변길이 레코드 페이지로 구분된다. 맵 데이터의 경우 LineString, Polygon 객체는 좌표의 개수가 제한되어 있지 않기 때문에 가변길이 레코드 페이지로 분류된다. Short 가변길이 레코드 페이지의 경우 페이지 내의 슬롯의 addressing이 2byte를 초과하지 않을 경우 슬롯 할당 비트맵의 크기를 최소화 하기 위하여 별도로 추가 되었다. 하나의 페이지를 넘는 Big Object의 경우 오버플로우 처리를 수행하며, 각 레코드 필드의 값의 범위에 따라 필드의 사이즈를 최적화하는 역할을 수행한다.

Access Path Manager: 접근 경로 관리자는 일반적으로 색인 관리자와 동일하다. 색인 관리자는 DBMS에서 제공하는 다양한 색인의 생성/삭제 등의 관리를 수행한다. FUNs의 모바일 DBMS에서 제공하는 색인은 B+-tree, GRID 및 R*-tree 가 있다. 1차원 속성 데이터의 빠른 접근을 위하여 B+-tree를 이용하고, 2차원 공간 데이터의 검색을 위하여 GRID와 R*-tree를 이용한다. R*-tree의 경우 가장 성능이 좋은 공간 색인으로 널리 알려져 있다. 하지만 모바일 단말에서 R*-tree의 경우 검색의 성능은 뛰어 나지만 공간객체 삽입/삭제/변경의 경우 tree 전체를 재구성해야 하며, 이 경우, 플래시메모리에서 쓰기 비용이 크기 때문에 전체 성능이 떨어진다. 즉, Air 업데이트를 지원하는 내비게이션과 같은 응용에 적합하지 않다. GRID 색인의 경우는 Air 업데이트를 위한 응용에 R*-tree 보다 적합하다.

Catalog Manager: 카탈로그 관리자는 데이터베이스, 테이블스페이스, 테이블, 색인의 메타 정보를 관리한다. 해당 메타 정보는 사용자가 질의처리기를 통해서 접근이 불가능하며 시스템에 의해서만 접근이 가능하다.

Transaction Manager, Log Manager: 모바일 DBMS에서는 일반적으로 다중 사용자를 지원하지 않는다. 즉, 동시성 제어 기능 등은 지원하지 않는다. 따라서 한번에 하나의 트랜잭션만 수행되며, 트랜잭션들은 서로 겹쳐서 동작하지 않는다. 이러한 트랜잭션 처리시 시스템 fail일 발생하였을 경우 로그 정보를 이용하여 시스템 복구를 수행한다. 기본적인 트랜잭션 처리 원칙은 WAL(Write Ahead Log) 프로토콜을 준수하며, 로그 페이지 이미지를 이용하여 undo/redo 기능을 제공한다.

Query Processor: FUNs 모바일 DBMS에서의 질의 처리기는 API 수준으로 제공을 하며 사용자들은 API를 이용하여 데이터베이스, 테이블스페이스, 테이블, 색인, 레코드 생성/삭제 및 검색을 수행하게 된다. 이 부분에 대해서는 3.2절에서 자세히 살펴보도록 한다.

3.2 모바일 공간 DBMS 특징

3.2절에서는 FUNs 모바일 공간 DBMS의 사용자 측면에서의 특징을 자세히 설명한다. <표 1>은 FUNs 모바일 공간 DBMS의 특징을 요약하였다.

FUNs 모바일 공간 DBMS와 응용과의 인터페이스는 SQL-like한 API가 제공되며, 속성 데이터 검색의 경우는 일반 DBMS에서 제공하는 검색기능 이외에 한글 초성검색기능을 추가로 지원한다. 공간 데이터 검색은 windows 검색, k-nearest neighbor 검색, [18]의 공간 위상관계 연산 기능을 제공한다. 그리고, 경로계산을 위하여 노드와 링크 테이블 레코드의 물리적 레코드 ID를 이용한 접근 API를 제공한다. 이는 경로계산을 위한 노드/링크 레코드에 대한 successor의 구현을 색인을 이용한 각각 레코드의 논리적인 ID를 이용하여 검색할 경우 경로계산의 성능을 보장하지 못한다. 따라

〈표 1〉 FUNs 모바일 공간 DBMS의 특징

Function		FUNs Mobile DBMS
SQL/API		C/C++ API
DDL		Create / Drop Tablespace, Table, Index
DML		Insert, Update, Delete
Search		Attribute Search (Match, initial Sound) Spatial Search (Windows, K-NN, Relational Search) Network Search (Physical Rid Search)
Transaction/Recovery		Begin, Commit, Redo, Undo
Object		Table (General Table, Network Table) Access Path (B+-tree, GRID, R*-tree)
Data Type	General	char, varchar, int, uint, short, double, float, binary, date, time, blob
	Spatial	point, lineString, polygon
Storage Media		Disk, Flash Memory
Supported OS		Windows, WinCE (Windows Mobile, Pocket PC)

〈표 2〉 FUNs 모바일 공간 DBMS의 검색 연산자

Search	Operator	Operator 명	비고
속성검색	>	GT	보다 크다
	<	LT	보다 작다
	=	EQ	같다
	>=	GE	크거나 같다
	<=	LE	작거나 같다
	<>	NE	다르다
	Like	LIKE	일반 Like 문, 'key%'지원
	Like(초성)	LIKE2	한글초성검색
공간 검색	Window	WINDOW	Window query
	KNN	K_NN	k-nearest neighbor search
	Relation	CROSS	OGC Simple Feature Geometry 9-intersection model
		DISJOINT	
		EQUALS	
		OVERLAPS	
		TOUCHES	
		WITHIN	
		INTERSECTS	
		CONTAIN	

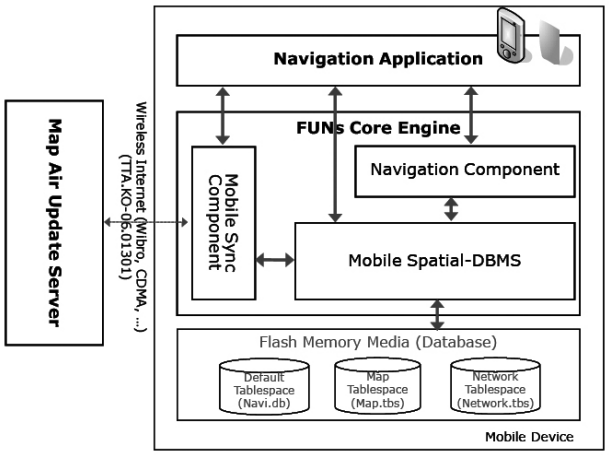
```
// examples for query execution
1.STRING select = _T("*"), from = _T("POI");
2.WHERE_CLAUSE where;

// window query
3.ADD_RESTRICTION(&where, _T("GEOMETRY"), WINDOW, 126.9, 37.5, 126.955, 37.542);
4.CResultSet *rs1 = pStmt->executeQuery(select, from, where, MAP_TBS_NAME);

// knn query
5.ADD_RESTRICTION(&where, _T("GEOMETRY"), KNN, 126.9, 37.5, 3);
6.CResultSet *rs2 = pStmt->executeQuery(select, from, where, MAP_TBS_NAME);

// initial sound query
7.ADD_RESTRICTION(&where, _T("NAME"), LIKE2, _T("ㅅㅇㅇ")); // 서울역 초성
8.CResultSet *rs3 = pStmt->executeQuery(select, from, where, MAP_TBS_NAME);
```

(그림 3) FUNs 모바일 공간 DBMS 질의 처리 API 수행 예



(그림 4) FUNs 시스템 구조

서 본 연구에서는 노드 레코드에 연결된 링크 레코드 정보와 링크 레코드의 양 끝에 연결된 노드 레코드 정보에 해당하는 물리적 레코드 ID 값을 할당하는 전처리 과정을 거친 후, 빠른 successor 검색을 제공한다. 공간 데이터 타입으로는 [18]의 Point, LineString, Polygon 자료 형을 제공한다. 검색기능에 대한 보다 상세한 내용은 <표 2>에서 요약하였다.

(그림 3)에서는 내비게이션에서 가장 많이 사용되는 연산자 중 Window, KNN, 초성검색의 질의 수행 예제를 나타내고 있다. 질의 대상 테이블은 POI 테이블이며, 이 테이블의 Point 공간 자료형의 필드 이름은 "GEOMETRY"이고 poi 명칭의 필드 이름은 "NAME"이다. 1, 2에서는 select, from, where 절에 해당하는 변수를 선언하고, 3과 4에서는 window 질의를 수행한 예이다. 의미는 사각형 영역 (126.9, 37.5), (126.955, 37.542)에 포함되는 POI 테이블 레코드 셋을 반환하는 예제이다. 5, 6에서는 위치 (126.9, 37.5)에서 가장 가까운 POI 테이블 레코드 3개를 반환하는 질의 예제이고, 7, 8에서는 "서울역"의 초성 "ㅅㅇㅇ"을 NAME 필드 값으로 가지는 레코드를 반환하는 예제이다.

4. 실시간 맵 업데이트 내비게이션 응용 구현

4장에서는 본 연구의 중요한 목적 중의 하나인 Air 업데이트가 가능한 내비게이션의 응용 구현을 위하여, 내비게이션 컴포넌트 및 Air 업데이트를 위한 모바일 동기화 컴포넌트에 대해서 살펴본다.

본 연구에서 개발한 시스템은 크게 서버(MAUS: Map Air Update Server)와 단말(FUNs: Flash-aware Ubiquitous

Navigation System)로 구성된다. MAUS는 Air 맵 업데이트를 위한 부분 맵 데이터(배경 데이터, POI, 도로 네트워크 데이터 등)를 제공하는 역할을 수행하며 이 부분에 대해서는 본 논문에서는 다루지 않는다. FUNs는 3장에서 기술한 모바일 공간 DBMS엔진과 이를 이용한 내비게이션 컴포넌트, 모바일 동기화 컴포넌트로 구성된다. 상위 경로 안내를 위한 응용의 구현은 FUNs Core 엔진 라이브러리를 이용하게 된다. 모바일 동기화 컴포넌트는 MAUS와 무선 통신(Wibro, CDMA 등)을 이용하여 부분 맵 데이터를 다운로드 받은 후 모바일 공간 DBMS에 반영하는 역할을 수행한다. 내비게이션 컴포넌트는 모바일 공간 DBMS를 이용하여 경로계산, 맵 매칭, 모의주행 등의 주요 API을 제공한다. 각각의 주요 구현 내용에 대해서는 4.1절과 4.2절에서 살펴보고자 한다.

4.1 모바일 동기화 컴포넌트

모바일 동기화 컴포넌트는 MAUS와 [19] 표준 프로토콜을 이용하여 데이터를 요청 및 처리한다. [19]의 XML 스키마에는 Air 맵 업데이트를 위한 MAUS와 단말간의 서비스와 이에 필요한 엘리먼트를 정의하고 있다. 표준에서 제시하고 있는 주요 서비스는 <표 3>과 같다.

Discovery 서비스는 단말 클라이언트가 업데이트 하고자 하는 지역에 대한 업데이트 데이터 요약 정보를 요청하고 응답 처리하는 서비스이다. 예를 들면, 업데이트 크기, 다운로드 예상시간 등의 정보를 요청할 수 있다. Provision 서비스는 업데이트 데이터를 요청하고 응답 처리하는 서비스이다. 이 두 가지 서비스는 기본적으로는 요청과 응답으로 이루어지지만 push 서비스 등을 위하여 Subscription 서비스를 정의하고 있다. 클라이언트 단말이 원하는 영역에 대한 필터 정보와 실제 업데이트 데이터 정보에 대한 엘리먼트가 표준 프로토콜에 정의되어 있다. 필터는 업데이트할 영역에 대한 정보를 나타내며, 파티션(Partition), 영역(Area), 객체(FeatRel) 등이 있다. 파티션은 일반적으로 관리의 목적으로 맵 데이터를 일정 셀(Mesh, Partition 등)로 나누게 되는데 필터의 단위를 이러한 파티션으로 정의를 한다. 영역의 경우 “서울”, “경기도” 등 명칭으로 명시를 하고 해당 영역에 대한 파티션과 매핑 시키게 된다. 객체의 경우 실제 객체 단위의 업데이트를 명시한다. 업데이트 데이터 엘리먼트는 실제 업

<표 3> MAUS-단말간 맵 Air 업데이트를 위한 서비스

Service	Desc
Discovery	Update Data 요약정보 요청/응답
Provision	Update Data 요청/응답
(Un)Subscription	Notification Service 설정 요청(해제)

데이트 데이터를 포함하는 자료구조이며 크게 ObjectData와 BinaryData로 구분 된다. BinaryData의 업데이트 단위는 해당 파티션의 바이너리 데이터 전체가 된다. 이 경우에는 업데이트된 객체가 일부임에도 불구하고 해당 파티션 바이너리 전체를 업데이트해야 하는 단점이 있으며, 이를 위하여 업데이트 된 객체만을 전송하는 ObjectData 자료구조가 정의되어 있다. (그림 5)에는 Partition ID 1234에 해당하는 데이터 중 version 2008-05-01T17:00:00 이상의 업데이트 데이터를 요청하는 Provision 서비스 요청 XML 예제이다.

4.2 내비게이션 컴포넌트

내비게이션 컴포넌트는 모바일 공간 DBMS를 이용하여 경로계산, 맵 매칭 등의 기능을 제공한다. 기존의 상용 내비게이션 시스템은 모바일 단말기에 최적화된 파일 시스템 형태로 맵 및 네트워크 데이터를 관리하고 이를 이용하여 경로계산을 하였다. 경로 계산 알고리즘에 대해서는 많은 연구가 진행되었고 모바일 단말에서 경로계산에 가장 많이 사용되는 알고리즘은 [20]의 Dijkstra 알고리즘과 A* 알고리즘, 양방향 Dijkstra 알고리즘, 양방향 A* 알고리즘 등이 있다. 기존의 내비게이션 시스템은 이러한 경로 계산 알고리즘에 몇 가지 성능을 높일 수 있는 방법을 이용하고 있다. 가장 일반적으로 이용되는 방법은 멀티레벨 네트워크 데이터 활용과 파티션 단위의 검색 영역 제한 등의 방법을 이용한다. 이 방법들을 모바일 DBMS를 이용할 경우 구현 방법들이 달라져야 한다. 가장 큰 차이점은 기존의 파일 시스템의 경우 노드/링크 데이터의 참조가 각각 파일 오프셋 정보를 이용하여 생성되고 이를 이용하여 각 데이터로의 직접적인 접근이 가능하지만, DBMS의 경우 ID를 이용한 검색은 비록 색인이 생성되어 있다 하더라도 네트워크 데이터의 레코드 개수가 많기 때문에 사용자의 요구사항을 만족할 만큼

```
<?xml version="1.0" encoding="UTF-8"?>
<MAUS-Terminal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=" MAUS-Terminal Protocol.xsd"
QueryType="REQUEST" ServiceType="PROVISION">
        <BaselineMap></BaselineMap>
        <Filter>
                <Partition>
                        <PartitionID>1234</PartitionID>
                        <Version>2008-05-01T17:00:00</Version>
                </Partition>
        </Filter>
</MAUS-Terminal>
```

(그림 5) Provision 요청 XML 샘플

의 성능 결과를 보장할 수 없다. 이에 이번 절에서는 모바일 공간 DBMS을 이용하여 경로 계산 기능을 어떻게 구현하였는지를 살펴보고자 한다.

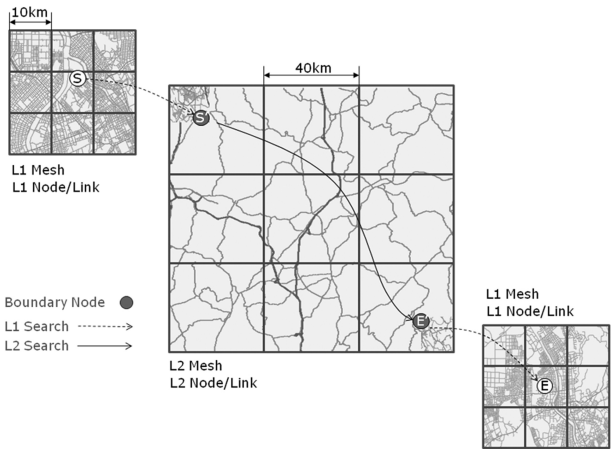
① 물리적 레코드 ID 접근 API

모바일 DBMS를 이용하여 도로 네트워크의 노드/링크 successor (Node에 연결되어 있는 링크와 링크에 연결되어 있는 노드를 반복적으로 찾는 함수)는 query-by-query를 이용하여 구현 할 수 있다. 이 경우 ID 값을 이용한 질의 수행의 반복은 시스템 성능을 저하시킬 수 있다. 이에 데이터 로더에서 노드/링크 테이블을 생성할 때 노드와 링크 레코드에 참조 물리적 레코드 ID (물리적 페이지 번호, 슬롯 번호로 구성)를 구축한 후 이 값을 이용하여 레코드를 직접 접근 가능한 API를 제공함으로써 successor의 성능을 보다 높일 수 있다. 내비게이션 컴포넌트에서는 물리적 레코드 ID의 접근 API를 이용하여 GET_NODE_RECORD_RID, GET_LINK_RECORD_RID, GET_BOUNDARYNODE_RECORD_RID, GET_LINKATTR_RECORD_RID, GET_NODEATTR_RECORD_RID 등의 매크로를 구현한 후 경로계산 알고리즘에서 이용한다.

② 멀티레벨 네트워크 데이터 테이블

내비게이션 컴포넌트에서는 도로 네트워크 데이터를 2레벨로 생성하여 활용한다. 즉, 레벨 1 데이터에는 모든 종류의 도로가 포함되어 있고, 레벨 2 데이터에는 일반도로를 제외한 주요 도로가 포함되어 있다. 장거리 경로계산의 경우 출발점에서의 boundary 노드, 도착점에서의 boundary 노드를 검색한 후, 출발점에서 출발점 boundary 노드 경로 계산, 출발점 boundary 노드에서 도착점 boundary 노드 경로계산, 도착점 boundary 노드에서 도착점 경로계산을 수행함으로써 최적의 경로를 찾게 된다[21]. (그림 6)은 장거리 경로의 경우 멀티레벨 경로계산 예를 나타내고 있다.

레벨 1 네트워크 데이터를 이용하여 레벨 2 네트워크 데이터를 생성하는 방법 중, 불필요한 노드의 삭제와 링크의 결합 등의 클린 작업이 필요하다. 이러한 클린작업에 의해 레벨 2 노드/링크 데이터 개수가 최소화하여 성능을 향상시킬 수 있다. 하지만 이럴 경우 네트워크 데이터의 업데이트 절차가 복잡하여 업데이트 성능이 저하될 수 있다. 이에 FUNs에서 레벨 2의 네트워크 데이터 생성 시에 클린 작업을 생략하였으며 그럼에도 불구하고 경로계산의 성능은 보장된다. 경로계산을 위한 레벨1, 레벨 2에 해당하는 테이블은 L1LINK, L1NODE, LINKATTR, NODEATTR, L2LINK, L2NODE, BOUNDARYNODE가 있다.



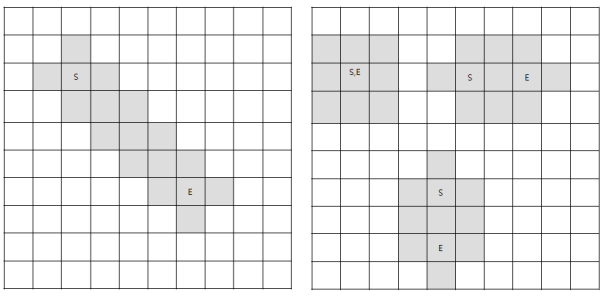
(그림 6) 장거리 멀티레벨 경로계산 예시

③ 검색 영역의 제한

기존 내비게이션 맵 데이터의 경우 동일 크기의 파티션을 생성하고 파티션 단위의 노드/링크를 저장 관리한다. 그리고 이 파티션정보는 경로계산의 검색 영역 제한 단위로 활용된다. 출발점과 도착점의 경로가 포함될 것으로 예상되는 파티션들을 추출하고 여기에 포함되어 있는 네트워크 데이터를 메모리에 적재한 후 경로계산을 수행하는 방법이 일반적인 방법이다. 이는 기존 파일 시스템에 저장되어 있는 단위가 파티션 단위이므로 이를 메모리에 적재하기가 용이하고 메모리에서의 경로계산이 성능을 보장할 수 있다. 모바일 DBMS에서는 이 방법을 이용하기 위해서는 각 파티션에 포함되어 있는 노드/링크 데이터에 대해서 별도 테이블을 구성할 수 있지만, 이 경우에는 테이블의 개수가 많아져서(수천 개 테이블) 성능이 저하될 수 있다. 또한 예상 파티션의 테이블의 모든 레코드를 메모리에 적재하는 것 또한 제한적인 리소스의 효율적인 사용 측면에서 효과적이지 못하다. 본 연구에서는 이런 방식을 따르지 않고 예상 파티션 ID 셋을 추출한 후 ①, ②의 동작에서 해당 노드/링크 레코드가 파티션 ID 셋을 참조하지 않는 경우에는 successor을 확장하지 않도록 검색 영역을 제한한다. (그림 7)은 본 경로계산 방법에서 사용하는 파티션 단위의 검색영역 제한 예시를 나타내고 있다.

4.3 내비게이션 응용 구현

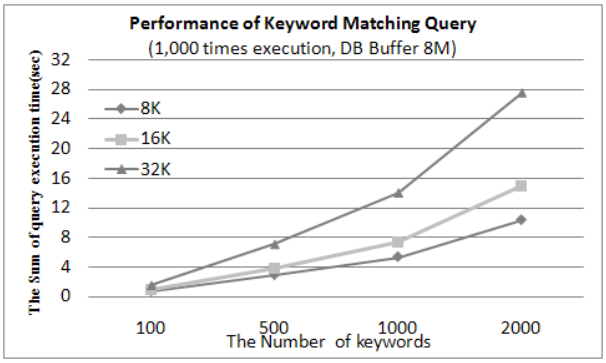
본 연구의 FUNs Core 엔진 라이브러리를 이용하여 내비게이션 응용을 구현하였다. 주요 기능은 일반 내비게이션과 유사하게 경로검색 및 경로안내 등의 기능에 Air 업데이트 기능을 추가 구현하였다. (그림 8)에서 (a)는 4.1의 모바일 동기화 컴포넌트를 이용한 디스커버리 서비스의 수행 화면이다. (b)는 provision 서비스를 수행하기 전/후 경로계산 결과가 달라지는 화면을 나타내고, (c)에서는 provision 서비스



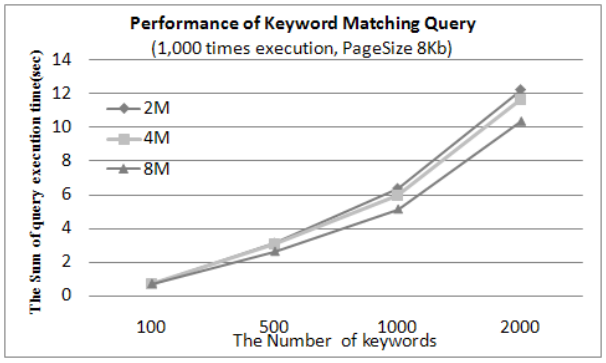
(그림 7) 검색 영역 제한 예

(a) (b) (c)

(그림 8) 내비게이션 응용 구현 화면



(a) 페이지 크기에 따른 성능



(b) 버퍼 크기에 따른 성능

(그림 9) 속성검색 실험 결과 측정

를 이용하여 표출용 공간 객체 업데이트 전/후 화면을 보여주고 있다.

5. 실 험

5장에서는 실험을 통한 FUNs의 성능 분석에 대하여 기술한다. 실험 항목은 크게 모바일 공간 DBMS의 검색 성능과 내비게이션 컴포넌트의 성능, 그리고 업데이트의 성능에 대해서 실험하였다. 실험에 이용된 데이터는 <표 4>와 같다. 전국 맵 데이터를 모바일 DBMS로 변환 및 로딩하였으며 데이터 파일은 두 개의 테이블스페이스 파일과 메타 등의 정보를 관리하는 한 개의 DB 파일로 구성된다. 데이터 종류는 화면 표출용 데이터, 검색용 POI 데이터 및 경로계산용 네트워크 데이터로 구성된다.

테스트 단말기는 LG KC1단말기이며, CPU 800 Mhz, 128M DRAM, 2G 플래시메모리를 이용하였다.

〈표 4〉 실험 데이터

Kind	Size(Mb)	Desc	File
Display	400	68 Tables, L1 ~ L6	Map.tbs
POI	300	1 Tables, 500,000 records	
Network	400	9 Tables, NODE - 600,000 records, LINK - 700,000 records	Network.tbs

5.1, 5.2절에서는 모바일 공간 DBMS의 검색 실험을 수행하였고, 5.3절에서는 경로계산, 5.4절에서는 업데이트 실험을 수행하였다.

5.1 속성검색 실험

속성검색 실험은 검색용 POI 테이블에 대하여 키워드 matching 검색(<표2>의 LIKE 연산)에 대해서 수행하였다. POI 테이블의 문자열 NAME 필드에 대하여 B+-tree 색인을 구축한 후 질의 대상 키워드를 100, 200, 1,000, 2,000개를 무작위 추출한 후 실험하였다. 실험에서의 주요 파라미터는 DB의 버퍼의 크기와 페이지 크기이다.

(그림 9)는 실험 결과 그래프이다. 세로축은 각각의 키워드를 검색한 시간의 합을 나타내고 가로축은 검색 데이터 셋의 크기를 나타내고 있다. 실험 결과 (a)는 페이지 크기 8K, 16K, 32K로 각각 데이터를 구축한 후 검색한 시간을 측정한 것이다. 그래프에서와 같이 8K로 페이지를 구축한 경우에 성능이 제일 좋으며 2,000개의 키워드를 검색하는데 소요되는 시간은 약 10초 정도이다. 실험 결과 (b)는 버퍼 관리자에서의 버퍼 크기를 2M, 4M, 8M로 설정한 경우 검색한 시간을 측정한 그래프이다. 8K의 페이지 크기에 8M의 버퍼 크기를 이용하는 경우에 성능이 가장 좋다.

5.2 공간검색 실험

공간검색 실험은 속성검색 실험과 마찬가지로 POI 테이블

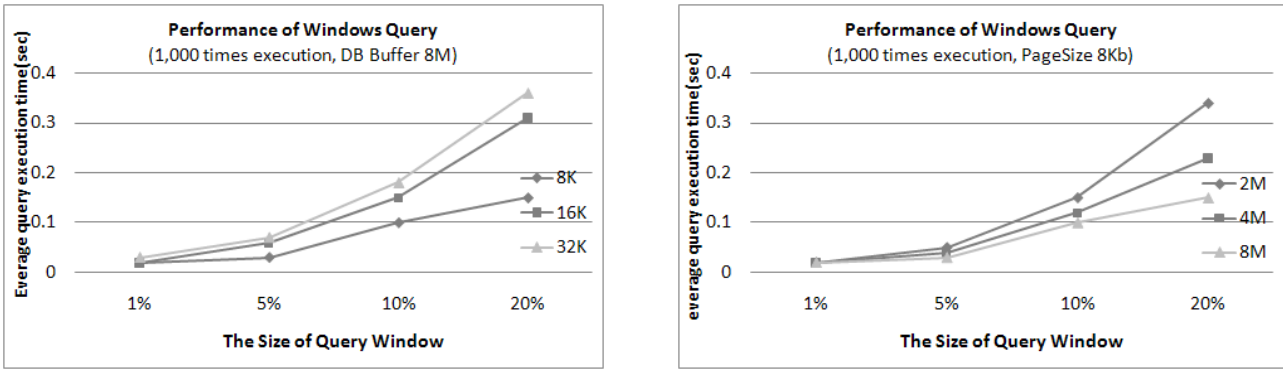
의 공간 객체 정보에 대하여 window 검색 시간을 측정하였다. POI 테이블의 필드 중에서 point 타입의 GEOMETRY 필드에 대하여 공간 색인 GRID를 구축한 후 질의 영역의 크기에 따른 검색 성능을 측정하였다. 질의 영역 크기는 (그림 10)에서와 같이 우리나라 전역 공간의 영역 1%, 5%, 10%, 20%에 대한 1,000개를 무작위로 생성한 후 평균 검색 시간을 측정하였다. 마찬가지로 DB의 페이지 크기와 버퍼 크기에 따른 성능을 비교하였다.

(그림 11)은 실험 결과 그래프이다. 세로축은 평균 검색 시간이고 가로축은 검색 영역의 크기이다. 실험 결과 (a)에서와 같이 8K로 페이지를 구축한 경우에 성능이 제일 좋으며 우리나라 전체에서 20% 영역에 해당하는 평균 검색 시간은 0.2초 정도이다. 실험 결과 (b)에서는 버퍼의 크기가 8M가 일 때 성능이 제일 좋다. 이 이외에 성능에 영향을 미치는 요소는 공간객체가 실제 물리적 페이지랑 얼마나 클러스터링이 잘 되어 있느냐와 데이터의 공간 분포에 따른 GRID 색인의 셀 크기를 어느 정도로 설정하여 생성하느냐 등이 있다.

5.3 경로검색 실험

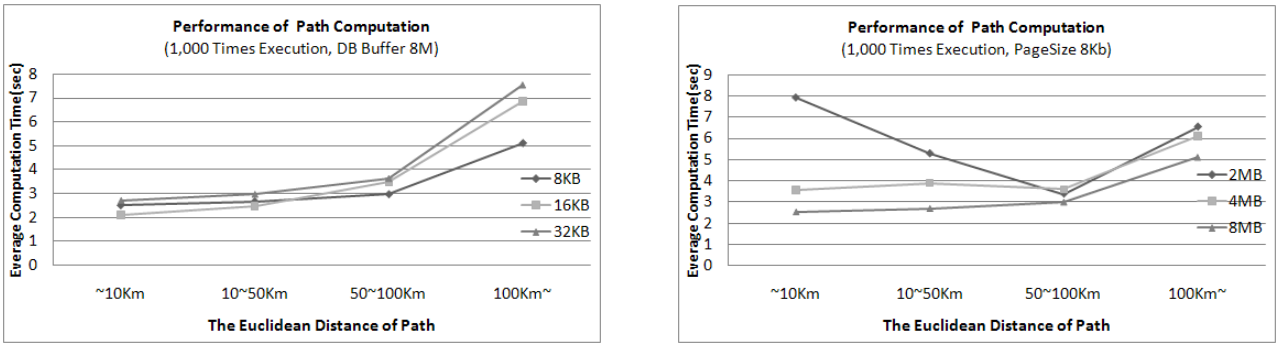
이번 절에서는 FUNs의 모바일 공간 DBMS를 이용한 경로 검색의 성능을 측정하였다. 테스트 경로는 직선거리 10Km 미

(그림 10) 우리나라 전체 공간에서의 공간질의 분포 1%, 5%, 10%, 20% 무작위 추출



(a) 페이지 크기에 따른 성능 (b) 버퍼 크기에 따른 성능

(그림 11) window 검색 실험 결과 측정



(a) 페이지 크기에 따른 성능 (b) 버퍼 크기에 따른 성능

(그림 12) 경로 검색의 실험 결과 측정

만, 10~50Km, 50~100Km, 100Km 이상에 해당하는 각각의 경로의 출발점, 도착점을 무작위로 1,000개를 추출한 후 평균 수행시간을 측정하였다. 여기서, 10Km 미만의 단거리의 경우에는 레벨 1 네트워크에서만 경로 검색을 수행하고, 이상의 거리에 대해서는 레벨 1, 레벨 2 네트워크 데이터를 함께 이용하여 경로 검색을 수행하도록 하였다. (그림 12)는 실험 결과 그래프이다. (a), (b)에서와 같이 페이지 크기 8K, 버퍼 크기 8M일 때 성능이 제일 좋으며 장거리(100Km 이상)의 경우 경로탐색 시간이 평균 5초 정도의 성능을 보인다. 10Km 미만의 경로 검색을 이용하는 경우, 레벨 1 네트워크 데이터를 이용하기 때문에 서울이나 광역시의 경우 데이터가 밀집되어 있는 지역에서는 장거리보다 성능이 더 떨어질 수도 있으며, (b)에서와 같이 2M 버퍼를 이용할 경우 실제 페이지 I/O가 많이 발생하여 성능이 더욱더 저하될 수도 있다.

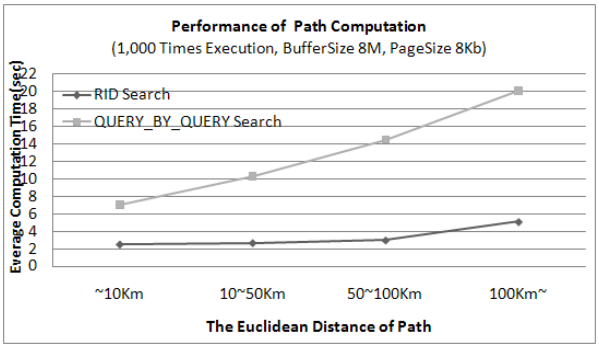
(그림 13)은 물리적 레코드 ID와 query-by-query 방법에 의한 경로계산 성능을 비교한 결과이다. 물리적 레코드 ID를 이용함으로써 일반적인 query-by-query 방법에 비해 약 4배 정도 성능이 향상됨을 알 수 있다.

FUNs를 이용한 경로탐색 성능과 기존 내비게이션의 경로계산 성능을 비교하였다. 성능을 비교하기 위하여 동일한 단말기(삼성전자 M4500)를 이용하였으며, 우리나라에서 가장 긴 경로에 가까운 서울-부산, 강릉-목포에 대한 경로와 데이터가 가장 많이 밀집되어 있는 서울 중심의 경로에 대한 계산 성능을 비교하였다. <표 5>는 성능 비교 결과를 나타내고 있으며, 부분 업데이트가 가능한 모바일 DBMS를 이용한 경로계산의 성능이 상용 제품인 A사 내비게이션의 성능과 유사함을 알 수 있다.

경로 계산 성능 이외의 다른 부분 성능 비교에 대해서는 향 후 연구과제로 남긴다.

5.4 맵 업데이트 실험

이번 절에서는 무선통신(CDMA, Wibro)을 이용하여 실제 맵 데이터 업데이트를 테스트하였다. 업데이트 데이터 셋은 실제 2007년 광주 지역 및 동탄 신도시 업데이트 데이터를



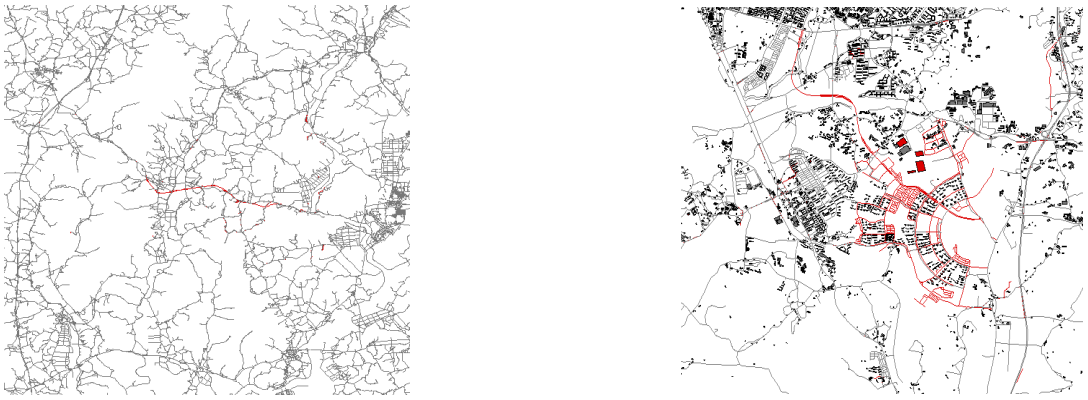
(그림 13) RID와 Query-by-Query에 의한 경로계산 성능 비교

〈표 5〉 FUNs와 A사 제품의 경로계산 성능 비교

Path	FUNs	A사 내비게이션
서울역-부산역	18초	22초
강릉역-목포역	12초	21초
강남역-서울역	6초	9초

이용하였다. (그림 14)는 광주 및 동탄 업데이트 데이터를 나타내고 있다. 광주 지역의 경우 도로 네트워크에 대한 업데이트 데이터만 존재하고, 동탄 신도시 지역에는 도로 네트워크 데이터와 배경 데이터가 존재한다. 그림에서 붉은색으로 표시 된 부분이 업데이트 객체이다.

보다 자세한 업데이트 데이터에 대한 설명은 <표 6>과 같다. 광주데이터의 경우 Node/Link 삭제 49건, 추가 264건, 변경 89건으로 총 업데이트 데이터 크기는 40Kbyte이다. 이때 무선통신을 이용한 전송시간을 제외한 FUNs 데이터베이스 업데이트 시간은 12초이다. 그리고 동탄 신도시 업데이트 데이터의 경우 Node/Link 삭제 107건, 추가 1,103건, 변경 124건으로 총 100Kbyte이며 업데이트 시간은 20초, 배경 공간데이터의 경우 추가 객체 2,532건이며 데이터 크기는 130Kbyte, 업데이트 시간은 8초가 소요되었다. 네트워크 업데이트의 경우 서버에서 전송하는 데이터는 레벨 1 네트워크 업데이트 데이터만을 전송하고, 모바일 동기화 컴포넌트에서



(a) 광주 지역 업데이트 데이터 (b) 동탄 지역 업데이트 데이터

(그림 14) 업데이트 맵 데이터

〈표 6〉 업데이트 맵 데이터 및 DB 업데이트 시간

Update Area	Update Data	DB Update Time
광주	Network 40Kb (Del: 49, Add: 264, Chg:89)	12 sec
동탄 신도시	Network 100 Kb(Del: 106, Add:1,103, Chg: 124)	20 sec
	Geometry 130Kb (Add: 2,532)	8 sec

상위 레벨 2 데이터를 생성하는 부분과 노드/링크 레코드의 물리적 레코드 ID 필드 값을 생성하는 부분이 추가되기 때문에 배경 객체 업데이트에 비해 시간이 많이 소요된다.

6. 결 론

본 논문에서는 무선통신을 이용한 맵 데이터의 부분 업데이트가 가능하도록 하기 위한 접근 방법으로 모바일 공간 DBMS를 개발하였고, 이를 이용한 내비게이션을 개발하였다. 앞서 살펴본 실험에서와 같이 부분 맵 업데이트가 가능한 모바일 DBMS의 성능은 실제 내비게이션에 적용될 만큼 뛰어난 성능을 보이는 것을 확인하였고, 또한 가장 중요한 경로계산의 경우에도 모바일 DBMS를 이용하여 성능에 전혀 문제가 없이 구현할 수 있음을 확인하였다. 이에 본 연구의 결과물을 활용함으로써 Air 업데이트가 가능한 내비게이션 서비스에 직접 활용이 가능할 것으로 기대된다. 또한 내비게이션 뿐만 아니라, LBS (Location-based service), 텔레매틱스, 모바일 GIS등 다양한 모바일 응용에서 활용이 가능하며, 모바일 환경에서의 효율적인 데이터 관리와 빠른 응용 개발이 가능할 것으로 기대된다. 향 후 연구과제로는 본 연구 결과물인 FUNs와 기존 내비게이션과의 성능을 추가로 비교 분석하는 실험을 수행하고, FUNs의 성능을 보다 향상시키기 위하여 실제 플래시메모리의 물리적 특성을 고려하여 OS에서 제공하는 파일 시스템과 플래시메모리 드라이버 등을 직접적으로 제어하도록 추가 연구하는 것이다.

감사의 글(Acknowledge)

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

참 고 문 헌

[1] EPEL, U. Grenoble, INRIA-Nancy, INT-Evry, U.Montpellier, U.Paris, U. Versailles, “Mobile Databases: a Seleciotn of Open Issues and Research Directions,” SIGMOD Record, Vol.33, No.2, pp.78-83, June, 2004

[2] Anil Nori, “Mobile and Embedded Databases,” In Proc. Of ACM International Conference on Management of Data, 2007.

[3] Sang-won Lee, Gap-Joo Na, Jae-Myung Kim, Joo-Hyung Oh, Sang-Woo Kim, “Research Issues in Next Generations DBMS for Mobile Platforms,” Mobile HCI, ACM September, 9-12, 2007.

[4] Anciaux N., Bouganim L., Pucheral P., “Memory Requirements for Query Execution in Highly Constrained Devices,” Int. Conf. on Very Large Data Base(VLDB), 2003.

[5] Gye-Jeong Kim, Seung-Cheon Baek, Hyun-Sook Lee, Han-Deok Lee, Moon Jeung Joe, LGeDBMS : a small DBMS for embedded system with flash memory, Proceedings of the 32nd international conference on Very large data bases, September, 12-15, 2006, Seoul, Korea

[6] E. Giguere, “Mobile Data Management: Challenges of Wireless and Offline Data Access,” Int. Conf. on Data Engineering (ICDE), 2001.

[7] Oracle Corporation, Oracle 9i Lite - Oralce Lite SQL Reference, Oracle Documentation, 2002.

[8] SQLServer for Windows CE “A Database Engine for Mobile and Embedded Platforms,” Proceedings of the 16th Internaitonal Conference on Data Engineering, P.642, February 28-March 03, 2000

[9] J. S. Karlsson, A. Lal, C. Leung, T. Pham, “IBM DB2 Everyplace: A Small Footprint Relational Database System,” Int. Conf. on Data Engineering (ICDE), 2001.

[10] Yun, J.K., Kim, J.J., Hong, D.S., and Han, K. J., “Development of an Embedded Spatial MMDBMS for Spatial Mobile Devices,” Proc. Of the 5th International Workshop on Web and Wireless Geographical Information Systems, (2005) 1-10.

[11] “Input for ISO Physical Storage Format - DRAFT v1.0 1997-04-04” (KIWI consortium, 1997)

[12] “KIWI+ Format edition 0.74” (Committee of KIWI+, 2001), http://www.kiwiplus.jp.

[13] ISO/DIS 14825, Intelligent transport systems - Geographic Data Files(GDF) - Overall data specification

[14] J.B. Bullock, E.J. Krakiwsky, “Analysis of the Use of Digital Road Maps in Vehicle Navigation,” Position Location and Navigation Symposium, 1994., IEEE, 11-15 April, pp. 494-501, Las Vegas, NV.

[15] http://www.etrico.com/en/activities/activities/actimap.htm

[16] Y.J. Joo, J.Y. Kim, S. H. Park, “Design and Implementation of Map Databases for Telematics and Car Navigation Systems using an Embedded DBMS,” The Journal of GIS Association of Korea, Vol.14, No.4, pp.379-389, December, 2006.

[17] http://www.internavi.ne.jp/
[18] OpenGIS Consortium, Inc., "The OpenGIS Simple Feature Specification for OLE/COM Revision 1.1," 1999.
[19] MCP-MAUS간서비스 프로토콜(TTA.KO-06.0129), MAUS-단말간 서비스 프로토콜(TTA.KO-06.0130), www.tta.or.kr.
[20] 이재무, 김종훈, 전홍석, "차량 항법 시스템의 경로 탐색을 위한 탐색 알고리즘들의 성능 비교," 한국정보교육학회논문집, 제2권 제2호, pp.252-259, 1998
[21] G. R. Jagadeesh, T. Srikanthan, and K. H. Quek, "Heuristic Techniques for Accelerating Hierarchical Routing on Road Networks," IEEE Trns. Intelligent Transportation Systems, Vol.3, No.4, pp.301-309, 2002.

민 경 욱
e-mail : kwmin92@etri.re.kr
1996년 부산대학교 전자계산학과(학사)
1998년 부산대학교 전자계산학과(석사)
2008년~현 재 충남대학교 컴퓨터공학과 박사과정
2001년~현 재 ETRI 텔레매틱스연구부 선임 연구원
관심분야 : GIS, LBS, 데이터베이스

안 경 환
e-mail : mobileguru@etri.re.kr
1997년 부산대학교 컴퓨터공학과(학사)
1999년 부산대학교 컴퓨터공학과(석사)
2004년 부산대학교 컴퓨터공학과(박사)
2004년~현 재 ETRI 텔레매틱스연구부 선임연구원
관심분야 : GIS, LBS, 데이터베이스

김 주 완
e-mail : juwan@etri.re.kr
1993년 부산대학교 컴퓨터공학과(학사)
1995년 부산대학교 컴퓨터공학과(석사)
2004년 충남대학교 컴퓨터과학과(박사)
1995년~현 재 ETRI 텔레매틱스연구부 팀장
관심분야 : GIS, LBS, Telematics

진 성 일
e-mail : sijin@cnu.ac.kr
1978년 서울대학교 계산통계학과(학사)
1980년 한국과학기술원 전산학과(석사)
1994년 한국과학기술원 전산학과(박사)
1987년~1989년 Northwestern대학 전산학과 객원교수
1983년~현 재 충남대학교 전자정보통신공학부 교수
관심분야 : 데이터베이스, 멀티미디어시스템, 소프트웨어공학