PostgreSQL/PostGIS 기반의 궤적 정보 저장 및 질의

Storing and Querying Trajectory Information on PostgreSQL/PostGIS

양 평 우[*] 이 용 미[**] 이 연 식[***] 남 광 우[****]
Pyoung Woo Yang Yong Mi Lee Yon Sik Lee Kwang Woo Nam

요 약 이 논문은 PostgreSQL/PostGIS 기반의 궤적 정보 저장과 질의에 대하여 기술하고 있다. 최근 모바일 단말 기술의 발전과 함께 위치기반서비스와 이동 객체 궤적에 관련된 많은 연구들이 진행되고 있다. 궤적은 이동 객체가 시간에 따라 변하는 위치정보들의 모음이며, 위치기반서비스를 위한 가장 중요한 정보중 하나이다. 기존의 공간 데이터베이스 시스템은 이동 객체 데이터 타입을 지원하지 않는다. 이 논문에서는 공간 데이터베이스로 많이 활용되고 있는 PostgreSQL/PostGIS 상에서 궤적 데이터 타입을 구현하고, 궤적 연산을 위한 궤적 질의 함수들을 제안하고 있다.

키워드 : PostgreSQL, 이동 객체, 궤적 관리, 위치기반서비스

Abstract This paper describes how to storing and querying trajectory information on PostgreSQL/PostGIS. Recently as technology of mobile devices is advancing, many researches for location-based services and moving object's trajectory have been studied. Trajectory is the set of information of the location by the time, and is one of the most important information for location-based services. Traditional spatial database systems do not support trajectory data types and functions. In this paper, we propose a trajectory data type and query functions for moving objects on PostgreSQL/PostGIS.

Keywords : PostgreSQL, Moving Object, Trajectory Management, Location Based Service

1. 서 론

최근 모바일 관련된 기술의 발전으로 스마트폰과 같은 다양한 모바일 단말이 보급되고 있다. 모바일 컴퓨팅 환경의 발달로 인하여 기존의 기술과 첨단 기술을 융합하여 새로운 서비스를 제공하려는 시도가 계속 되고 있다. 특히 GPS가 탑재된 모바일 단말 및 스마트 폰의 보급이 활발해지면서 위치정보 서비스(LBS)는 GIS의 공간 객체 정보와 GPS의 사용자 위치정보를 결합하여 다양한 서비스를 제공하고 있다. 예를 들어 교통 체증에 대한 정보를 실시간으로 보여주는 서비스, 네비게이션 서비스, 또한 어린 자녀들의 실시간 위치를 알려주는 서비스, 친구 찾기 서비스 등 실시간 위치 정보를 기반으로 하는 여러 서비스들이 제공되고 있다.

GIS와 GPS를 기반으로 하는 서비스의 대상이 되는 객체는 건물, 도로, 지역 같은 고정적인 위치의 위치정보를 갖고 있는 객체의 뿐만 아니라 자동차, 사람 같은 시간에 따라 위치가 변하는 이동 객체[6]도 포함이 된다. 기존의 공간 데이터베이스에서는 이동 객체의 위치와 시간을 따로 저장하여야 하기 때문에 기존의 기술이나 이론을 이동 객체에 직접적으로 적용하는 것은 부적절하다. 이동 객체를 처리하기 위해서는 기존의 공간데이터베이스에서 시

† 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. 2009-0067958)이며, 2010년도 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 결과임

[*] 군산대학교 컴퓨터정보공학과 석사과정 manner7979@kunsan.ac.kr
[**] 충북대학교 컴퓨터과학과 박사과정 ymlee@dblab.chungbuk.ac.kr
[***] 군산대학교 컴퓨터정보공학과 교수 yslee@kunsan.ac.kr
[****] 군산대학교 컴퓨터정보공학과 부교수 kwnam@kunsan.ac.kr(교신저자)

간에 대한 정보도 처리가 가능한 이동 객체 데이터베이스가 필요하다.

이동 객체 데이터베이스는 공간상에서 계속 변하는 사용자의 위치에 맞는 지리정보를 가져와서 데이터를 처리해야 하기 때문에 공간 데이터를 효율적으로 사용하여야 한다. 하지만 이러한 공간 데이터는 방대하기 때문에 모바일 단말기만으로 데이터 처리하기 힘들고 대용량의 공간 데이터베이스 안에서 저장하고 처리하여야 한다. 기존의 데이터베이스에 궤적을 저장하기 위해서는 이동 객체의 시간정보와 위치정보를 각각 저장을 한다. 이 때, 각 이동 객체의 한순간의 위치마다 테이블의 한 row에 저장을 하게 되고, 각 row마다 이동 객체에 대한 정보 또한 저장을 하여야 한다. 이 방법은 중복되는 데이터가 많이 저장되기 때문에 데이터베이스의 공간을 낭비하게 된다. 또한 row 수의 많은 증가로 인하여 검색의 효율성 또한 떨어지게 된다.

본 논문에서는 대용량의 공간 데이터베이스인 PostgreSQL에서 궤적을 저장 및 검색하기 위한 시스템을 제안하고, 이 시스템을 PostTrajectory시스템이라 명하였다. PostTrajectory에서는 이동 객체 타입인 tpoint를 제안하고 tpoint의 삽입/삭제/갱신/검색 기능을 구현 및 실험하였다. 본 논문에서 제안된 기법은 기존의 데이터베이스에서 timestamp와 point를 이용하여 저장하였을 때에 대비하여 저장의 효율성이나 검색의 효율성을 증대시킬 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 궤적 정보에 관련된 공간 객체와 이동객체에 관한 관련 연구를 소개하고, 3장에서는 trajectory 시스템의 설계를 소개한다. 4장에서는 시스템의 성능 분석을 하고 5장에서는 향후 연구와 결론을 맺는다.

2. 관련 연구

이 장에서는 이동 객체를 표현하기 위하여 기존의 공간 객체 모델과 이동객체 모델의 연구에 대해서 알아본다.

일반 데이터베이스에서는 연속성을 가지고 있는 이동 객체를 처리하기에는 부족한 면이 있다. 따라서 이동 객체에 대한 새로운 모델을 개발해야 하고 그에 따른 질의 처리 방법들을 개발하여야 한다. 이동 객체에 대한 연구인 CHROCHRONOS[1,7,13]에서는 이동점객체, 이동면 객체 등 공간 객체에 이동성을 부여한 이동 객체를 정의하고 각 데이터 형에 대해 각 연산자를 정의하였다. 또한 공간 객체의 기본 연산자들을 확장한 이동객체의 몇 가지 기본 연산자를 구현하기 위한 알고리즘을 제안하고 있다[2]. DOMINO(Database fOr MovINg Objects)[13]는 DBMS에 이동 객체 데이터베이스를 지원하는 것이 목적으로 연구되었다. DOMINO에서는 이동 객체가 업데이트 되면 현재 위치뿐 아니라 예상되는 미래 위치도 제공을 하기위해 MOST 모델을 제안하였다. 또한 FTL(Future Temporal Logic) 질의 언어를 제안하여 기본 SQL에 시간 연산자와 공간 연산자를 추가하였다.

오라클 10g같은 최신의 ORDBMS안에 패키지 형태로 들어가는 HERMES는 Trajectory Database를 위한 최신의 질의 처리 알고리즘을 완벽한 셋을 나타내는 첫 번째 연구 결과물이다[6,8,9,10]. HERMES는 오라클 10g의 객체 관계 데이터베이스 관리 시스템의 시공간 기능을 제공하기 위하여 시스템 확장을 통하여 개발되었다[10]. HERMES System의 주요 목표는 연속적인 이동 객체의 모델링과 질의의 지원이다. 이 시스템은 이론적인 시간이나 이론적인 공간 시스템을 통해 각각 사용될 수 있는 방법을 위해 설계되었고 중요 기능은 연속적인 이동 객체 질의와 모델링을 지원하는 것이다. HERMES는 데이터 타입의 집합과 그에 상응하는 연산자들이 정의되고, 오라클 데이터 카트리지를 통해 개발되고 제공되었다. HERMES의 ORDBMS 층에는 오라클 ORDBMS 서버에 궤적 데이터 저장과 LBS 지원을 위한 내부 구조의 질의 능력에 적합하게 강화되었다. HERMES-MDC(HERMES Moving Data Cartridge)는 이 ORDBMS층의 핵심으로 움직임의 집합 구축, 형상 축소나 확장 뿐만 아니라 다양한 시간의 기본적인 형태를 제공한다. 예를 들어 HERMES는 시공간 콘텐츠를 처리하는 통신 데이터 웨어하우스 상의 플러그인으로 사용할 수 있다.

이 논문에서는 궤적 저장을 위하여 HERMES시스템의 ORDBMS Tier와 같은 데이터베이스 시스템으로 PostgreSQL을 사용하였다. 시공간 데이터의 저장을 위한 tpoint 데이터 타입을 제안하고 tpoint를 이용한 데이터베이스의 연산을 설계 및 구현하였다. 다음 장은 PostTrajectory 시스템의 설계

에 대하여 설명을 할 것이다.

3. PostTrajectory 시스템의 설계

3.1 시스템 구조

PostTrajectory 시스템은 PostgreSQL에 궤적 데이터의 처리를 지원해주기 위하여 궤적에 필요한 데이터 타입과 각 데이터의 삽입, 삭제, 갱신, 검색 기능의 지원과 데이터를 이용한 여러 가지 응용에 사용할 수 있도록 Distance, Enter같은 이동 객체 연산도 포함하였다. 그림 1은 PostTrajectory 시스템의 전체적인 시스템 구조를 보여주고 있다. 사용자가 Web Client를 이용하여 자신의 위치와 시간정보를 서버로 보내게 되면 서버는 PostTrajectory기법을 이용하여 궤적 타입의 데이터로 변환 후 PostgreSQL 데이터베이스 저장을 하게 된다.

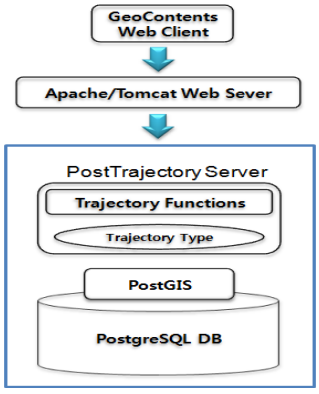

그림 1. 시스템 구조

그림 1에서 보이는 PostgreSQL Server는 PostgreSQL에 Trajectory를 삽입하기 위하여 본 논문에서 제안하고 있는 Trajectory저장 기법을 적용한 데이터베이스로 Hermes 시스템의 ORDBMS Tier와 같은 기능을 수행하는 서버이다.

3.2 궤적 데이터베이스 요소

PostgreSQL에서 지원되는 기본 데이터 타입에는 시간과 위치의 정보를 같이 저장할 수 있는 데이터 타입이 없기 때문에 본 시스템에서는 시공간 데이터를 나타낼 새로운 데이터 타입으로 그림 2와 같은 tpoint를 정의 하였다. 이 tpoint에는 객체의 움직임 정보를 저장할 point(GPS 데이터)와 데이터의 시간을 저장하게 된다.

```
CREATE TYPE tpoint(
      p        geometry,
      ptime    timestamp with timezone
);
```

그림 2. tpoint 데이터 타입

tpoint는 객체의 한순간의 위치 및 시간정보를 나타내기 때문에 궤적의 정보를 나타내기 위해 그림 3과 같은 mpoint를 정의한다.

tpoint : (p, t)

mpoint : $<(p_1, t_1), (p_2, t_2).....(p_n, t_n)>$

그림 3. mpoint의 정의

실제 제안한 시스템에서는 trajectory table에 mpoint를 저장하기 위하여 tpoint의 배열로써 궤적을 저장하였다.

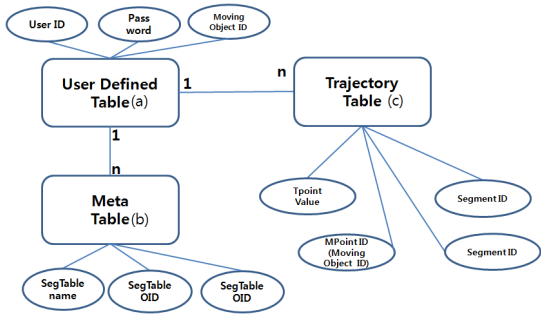

그림 4. 전체 테이블 관계도

이 시스템에 사용되는 테이블은 총 3종류로 구성되어 있다. 그림 4는 각테이블 간의 관계를 보여주고 있다. user defined table(a)는 사용자 정보를 저장하기 위한 테이블이다. 사용자의 정보를 저장할 수 있는 테이블을 생성한 후 이 테이블에 그림 5와 같이 trajectory_column을 추가하여 trajectory를 관리 하게 된다.

```
SELECT AddTrajectoryColumn('my_scheme',
'taxi', 'traj', 4236, 'MOVINGPOINT', 2, 3);
```

그림 5. trajectory 컬럼의 추가

그림 5의 addTrajectoryColumn()함수는 사용자 정보를 저장하는 테이블에 trajectory_column을 추가해주는 함수이다. trajectory_column은 해당 객체의 고유 moid와 객체의 trajectory를 저장할 테이블

의 oid를 갖는다. trajectory_segtable은 사용자 정의 테이블을 생성하면 자동으로 생성되는 테이블이다. 이 테이블에는 이동 객체가 이동한 궤적(trajectory)을 저장하게 된다. 각 이동 객체는 각각의 mpid (Moving Point IDentify)를 갖게 되고 각 mpid 마다 많은 수의 데이터가 삽입이 되므로 이를 관리하기 위하여 segid를 만들어 순서를 쉽게 구별 할 수 있도록 하였다. 하나의 segid에는 addTrajectory Column에서 지정해준 숫자(tpseg의 size)만큼의 위치 정보를 저장하게 된다. rect는 한 segid에 있는 위치들의 둘러싼 rectangle을 표현하기 위한 정보이다. 또한 next_segid와 before_segid를 만들어 각 segid를 double linked list처럼 관리를 할 수 있다.

마지막으로 meta table인 trajectory_columns 테이블이다. 이 테이블은 사용자가 지정한 trajectory_column의 이름과 자동 생성된 trajectory_segtable의 이름 등 각 테이블에 대한 전체적인 정보를 갖고 있는 메타 테이블이다.

3.3 궤적 SQL의 설계 및 구현

제안하는 시스템은 궤적 데이터 타입을 위해 사용자 정의 함수로 구현하였다. 사용자 정보의 저장 과 삭제를 위한 insert와 delete를 사용하는 방법과 이동 객체의 위치 정보를 저장하기 위한 append, remove, modify, select 함수를 설계하고 구현하였다.

그림 6은 사용자 정의 테이블에 데이터를 삽입하는 것을 보여주고 있다. 그림 6과 같이 사용자의 정보만을 삽입하면 trajectory_column에 자동으로 moid와 trajectory segment table의 OID가 등록이 된다. 이 trajectory_column을 이용하여 궤적정보를 삽입할 수 있다.

```
INSERT INTO taxi
VALUES(1, '57누2001', 'Optima', 'hongkd7');
```

그림 6. insert함수의 예

delete는 사용자 정의 테이블에서 사용자 정보를 삭제할 때 사용한다. delete시 사용자 정보의 mpid에 해당하는 데이터를 trajectory_segtable에서 삭제해 줘야 한다. 이를 위하여 trigger를 이용하여 delete가 사용될 때 사용자 정보의 삭제를 하기 전에 해당 사용자의 mpid에 해당되는 데이터를 삭제한 후 user_defined 테이블에서 사용자의 정보를 삭제해준다.

3.3.1 궤적 정보의 추가

궤적 정보의 추가는 append()함수를 사용한다. append는 사용자의 mpid를 알 수 있는 traj값과 입력할 수정할 tpoint값(위치값과 시간값)을 이용하여 데이터 삽입을 하게 된다. append를 이용하여 사용자가 이동 객체의 데이터 입력을 하게 되면 user_defined 테이블과 trajectory_columns테이블에 있는 데이터를 이용하여 입력한 이동 객체의 trajectory 데이터가 어느 trajectory_segtable에 입력이 되어야 하는지 찾고, 입력한 데이터가 입력될 trajectory_segtable의 mpid와 segid를 계산하여 삽입을 해줘야 한다. 또한 입력한 데이터가 기존에 있던 데이터보다 시간이 더 빠르다면 삽입을 할 수 없도록 처리를 해줘야 한다.

그림 7은 실제 append의 사용 예로 택시의 궤적을 추가하는 예제이다. 데이터의 삽입은 update를 이용하여 이루어지며 궤적 정보인 trajectory와 위치 및 시간(tpoint)을 입력하면 된다.

```
UPDATE taxi
SET traj = append(traj, tpoint(st_point(200 300),
        TIMESTAMP '2010-12-15 12:00:01+09'))
WHERE taxi_id = 5;
```

그림 7. append함수의 사용 예

그림 8은 append함수의 알고리즘을 보여주고 있다. trajectory_segtable에서 mpid를 검사(1)하여 mpid가 없다면(2) 최초 삽입이기 때문에 mpid, segid를 새롭게 생성하여 삽입을 한다.(3) mpid가 존재한다면, 해당 mpid의 row중에 next_segid가 null인 segid를 찾아서(4) 해당 segid에 tpseg의 사이즈를 최대 tpseg 사이즈와 비교(5)하여 삽입할 수 있는 여유가 있다면 삽입을 해주고 tpseg가 최대 사이즈와 동일하다면 새로운 segid를 생성하여 tpoint를 삽입한다. tpoint를 삽입 후 해당 row의 start_time, end_time, rect, mpcount등을 설정해준다.(6) 모든 저장 및 설정이 완료되면 처음에 받았던 trajectory 값을 리턴 하여, 해당 이동 객체의 궤적을 저장하게 된다.

3.3.2 궤적 정보의 삭제

궤적 정보를 삭제하기 위해서는 remove함수를

```
input
 traj : trajectory(이동 객체의 id정보)
 tpoint : TPOINT(삽입할 이동객체의 위치 및 시간정보)
output
 trajectory(이동 객체의 id정보)

begin
mpid = getmpid(traj);-------------------------------(1)
if mpid is null then--------------------------------(2)
    create new mpid, segid;
    insert tpoint ---------------------------------(3)
else
    segid = getNextSegidIsNull(mpid);---------------(4)
    if isFull(segid) then-----------------------------(5)
        create new segid;
        insert tpoint in new segid
    else
        insert tpoint in segid
    end if
end if
segid insert segInfo(startTime, endTime,
rect...)---------(6)

return trajectory
```

그림 8. append함수의 알고리즘

사용한다. remove는 사용자가 기존에 있는 궤적 데이터에서 어느 일정 부분만을 삭제해 줄 때이다. remove 명령을 사용하기 위해서는 삭제할 데이터의 traj값(mpid값)과 삭제하길 원하는 시작 시간과 끝 시간을 입력해주면 된다. remove는 해당 데이터를 삭제한 후 해당 row에 더 이상의 trajectory 데이터가 남아있지 않다면 해당 row를 삭제해주고 해당 row 이전 row의 segid와 이후 row의 segid를 알맞게 수정해 줘야 한다. 그림 9는 실제 remove함수를 사용하는 예이다.

```
UPDATE taxi
SET traj = remove(traj, TIMESTAMP '2011-01-
           10 13:35:07+09', TIMESTAMP '2011-01-
           15 13:40:15+09'))
WHERE taxi_id = 5;
```

그림 9. remove함수의 사용 예

3.3.3 궤적 정보의 수정

그림 10은 modify의 알고리즘을 보여주고 있다. modify의 경우에는 사용자 정보인 traj값과 수정하길 원하는 데이터의 시작 값, 끝 값과 tpoint array를 입력하여 준다. 이 경우에는 먼저 사용자가 수정하길 원하는 데이터의 시간과 사용자가 입력한 데이터의 시간의 일치하는지 확인을 하고 일치할 경우(1) 데이터 수정을 한다. 데이터 수정의 경우 한 row의 일정 부분 데이터가 삭제(2)가 되고 해당 row에 다 들어가지 않는 데이터가 삽입이 될 경우가 발생 할 수도 있는데(3) 이 경우에는 새로운 row를 생성(4)하여 나머지 데이터들을 삽입(5)해주고 segid를 알맞게 수정(6)해 줘야 한다. 수정도 삽입과 마찬가지로 trajectory 값을 리턴을 하면, 해당 이동 객체의 궤적을 수정을 하게 된다.

```
input
 traj : trajectory(이동 객체의 id정보)
 start_time : TIMESTAMP(수정하길 원하는 시작시간)
 end_time : TIMESTAMP (수정하길 원하는 끝시간)
 tpoint[] : TPOINT[] (수정될 이동객체정보 Tpoint)

output
 trajectory(이동 객체의 id정보)

begin
mpid = getmpid(traj);
if user's times contains tpoint[]'s times then----------(1)
   if segid's times contains user's times then
      while tpoint's times contains tpseg's time loop--(2)
         delete tpseg;
      end loof
      while tpseg_max_size > mpcount loo
         insert tpoint[i];
         mpcount++;
      end loof
      if remains tpoint----------------------------(3)
         create new segid;-------------------------(4)
         insert the remaining tpoint in new segid;-----(5)
      end if
   end if
end if
new segid insert segInfo(startTime, endTime, rect.....)--(6)

return trajectory
```

그림 10. modify 함수의 알고리즘

3.3.4 궤적 정보의 검색

select를 이용하여 다양한 질의가 가능하다. 본 논문에서는 select를 지원하기 위하여 slice, enter, leave, pass같은 함수들을 구현하였다. 표 1은 함수들에 대한 설명이다.

다음은 실제 질의를 어떻게 사용하는지에 대한 예제이다. 일정한 영역을 기준으로 그 영역에 있었던 이동 객체들의 궤적 검색질의의 경우 다음과 같이 사용할 수 있다.

그림 11의 질의는 영역과 이동 객체를 지정해주면 이동 객체가 해당 영역에 있었던 모든 궤적과 시간정보(tpoint[])를 리턴해준다.

표 1. select를 지원하기 위한 함수

함 수	설 명
slice(timestamp, timestamp)	시간을 기준으로 궤적을 분할해준다.
slice(geometry)	영역을 기준으로 궤적을 분할해준다.
enter(tpoint[], geometry)	trajectory가 해당영역에 들어갔는지 확인하는 함수이다.
leave(tpoint[], geometry)	trajectory가 해당영역에서 나왔는지를 확인하는 함수이다.
pass(tpoint[], geometry)	trajectory가 해당영역을 통과하였는지를 확인하는 함수이다.

```
SELECT slice(taxi.traj, 'Polygon(10 10, .....)')
FROM taxi;
```

그림 11. slice함수 질의 사용 예

그림 12는 pass함수와 slice함수를 같이 사용한 예제이다. slice 함수를 이용하여 궤적을 잘라내고 잘라낸 궤적이 지정한 영역을 통과했는지 확인하는 질의이다. 그림 12의 질의는 '아침 9시부터 오후 6시까지 군산대를 통과한 이동 객체를 보여주어라'와 같은 질의를 처리하기 위한 질의이다.

```
SELECT taxi_id, taxi_number FROM taxi
WHERE pass( (slice(taxi.traj, TIMESTAMP
                '2011....., TIMESTAMP '2011.....),
                polygon(10 10, .....));
```

그림 12. pass함수의 사용 예

4. 구현 및 성능 분석

이 장에서는 제안한 시스템의 삽입/삭제/갱신/검색기능을 테스트한 결과를 보인다. 제안한 시스템은 윈도우즈7 환경에서 PostgreSQL 8.4/PostGIS 1.4를 이용하여 구현되었다. 실험에 사용된 데이터는 Generator for network-based moving object[2]를 이용하여 생성한 데이터를 이용하였다. 실험은 Post-Trajectory의 segment_table에 저장되는 궤적정보의 크기에 따라 삽입/삭제/갱신의 시간을 비교하는 것으로 진행하였다. 실험을 위하여 500개의 이동 객체가 임의의 위치에서 이동하는 데이터를 생성하였으며 각 이동 객체마다 최소 50개에서 최대 1000개의 이동 위치의 데이터를 만들었다. 데이터 입력에 걸린 시간은 초(s) 단위이고, 데이터가 입력되는 테이블은 한 row에 입력이 되는 tpoint의 개수를 결정하는 tpseg의 size를 50, 80, 100, 150, 200, 250, 300으로 구분을 지어 각 size마다 삽입되는 시간, 삭제되는 시간, 수정하는데 걸리는 시간을 실험하였다.

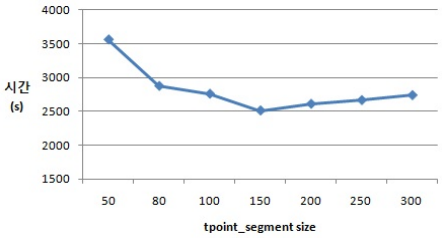

그림 13. append함수 실행 성능 시간

그림 13은 add 함수를 테스트한 결과이다. tpseg의 size를 50개로 했을 때의 시간이 가장 오래 걸렸으며 이후 80개를 했을 시에는 삽입시간이 큰 폭으로 줄어들고 80개 이후부터 100개, 150개씩 삽입을 할 때는 소폭 감소하다가 tpseg의 size가 150개일 때가 최소시간이 걸리고 그 후에는 다시 약간씩 증가하는 것을 보여준다. 이는 데이터베이스마다 한 페이지에 저장되는 데이터의 크기에 따른 다른 효과를 보여주는 것이라 예상이 되고, 데이터베이스에 따라 다른 결과를 보여줄 것이라고 예상된다.

그림 14는 remove 함수의 실행 성능을 실험한 결과이다. remove할 때는 각각의 테이블에서 최소 2개 이상의 row에 있는 데이터에 접근할 수 있도록 삭제되는 데이터의 양을 50, 80, 100, 150, 200, 250, 300개씩 삭제를 하였다. 삭제할 때 소요된 시간은 각각 삭제되는 데이터가 많을수록 더 많은 시간이 소요된다. 삭제시의 데이터도 삽입과 같이 데이터가 몇 페이지에 저장되어있는지에 따라 다른 결과를 보여준다고 볼 수 있다.

그림 15는 modify를 실행하여 걸린 시간을 측정한 결과. 그래프를 보면 tpseg의 size가 50일 때 가장 오래 걸리지만 그 이후는 거의 균일한 시간이

걸리는 것을 볼 수 있다. size가 50일 때의 시간은 jvm의 초기화 시간이나 기타 java의 특성 때문에 시간이 오래 걸린 것으로 예상이 되고 기타 다른 시간은 삽입과 비슷한 시간이 걸린 것을 확인할 수 있다.

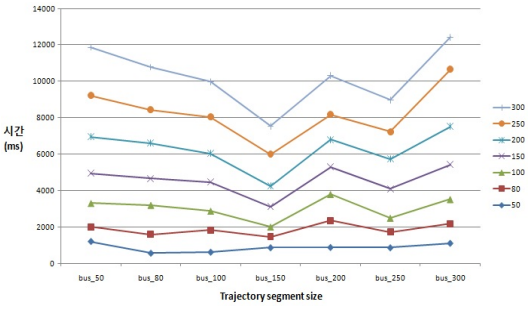

그림 14. remove함수 실행 성능 시간

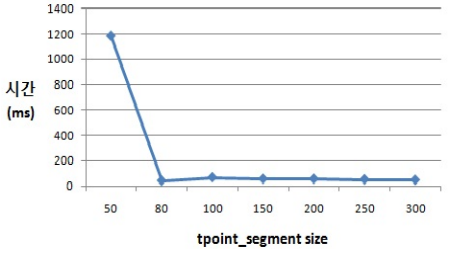

그림 15. modify함수 실행 성능 시간

그림 16은 검색 질의 중 slice의 검색 성능을 비교한 결과를 보이고 있다. 검색 범위는 데이터의 전체 시간에 비례하여 2%, 4%, 6%, 8%, 10%, 15%, 20% 씩 일정하게 검색 범위를 증가시켜 검색하도록 하였다. 그래프에서 보는 것처럼 검색 성능은 검색하는 데이터의 양에 따라 검색 시간도 비례하여 증가하는 것을 볼 수 있다.

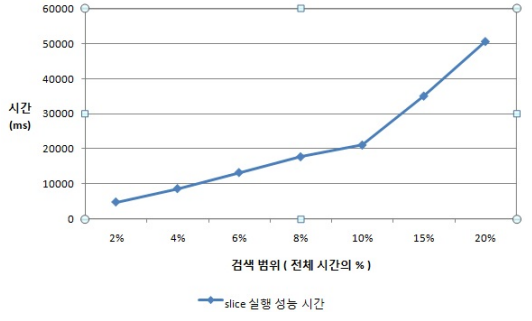

그림 16. slice함수 실행 성능 시간

5. 결 론

이 논문에서는 기존의 ORDBMS에서 이동 객체의 저장 및 관리를 위한 기법을 위하여 tpoint를 제안하고 tpoint의 삽입/삭제/갱신/검색 기법을 구현하였다. 제안된 기법은 기존에 개발된 OGC의 SFG를 기반으로 하였다. 구현된 System을 활용하면 스마트폰이나 PDA같은 이동 객체에서 보내지는 위치정보를 저장하고 다른 응용프로그램에서 이를 활용할 수 있다. 또한 단순히 현재의 위치를 기반으로 서비스를 제공하고 있는 기존의 LBS서비스에서 현재와 과거의 위치를 이용한 '군산대에 1시간동안 머물렀던 자동차를 보여주어라'와 같은 여러 서비스에 응용할 수 있다. 향후 연구 계획으로는 실제 서비스에서 예상할 수 있는 여러 질의에 관한 연구와 DB에서 빠른 검색을 위한 인덱스 구축 방안에 대한 연구가 필요하다.

참 고 문 헌

[1] T. Abraham, J. F. Roddick. 1999, "Survey of Spatio-Temporal Databases," GeoInformatica 3(1), pp. 61-99.

[2] T. Brinkhoff, 2000, "Generating Network-Based Moving Object.," Proceedings of the 12th international conference on Scientific and Statistical Database Management, pp. 253-255.

[3] M. Erwing, R. H. Güting, M. Schneider, M. Vazirgiannis, 1999, "Spatio-Temporal Data Type: An Approach to Modeling and Querying Moving Object in Databases," GeoInformatica 3(3), pp. 269-296.

[4] M. Erwig, M. S chneider, 1999, "Developments in Spatio-Temporal Query Languages," Proc. IEEE Int'l Workshop Spatio-Temporal Data Models and Languages, pp. 441-449.

[5] L. Forlizz, R. H Güting, E. Nardelli and M. Schneider, 2000, "A Data Model and Data Structures for Moving Object Databases," Proc. ACM SIGMOD, pp.319-330.

[6] R. H. Güting, M. Schneider, 2005, "Moving Object Databases," Morgan Kaufmann Publishers, CA.

[7] R. H. Güting, M. H. Bohlen, M. Erwig, C. S.

Jensen, N. A. Lorentzos, M. Schneider, M. Vazirgiannis, 2000, “A Foundation for Representing and Querying Moving Objects,” ACM Transactions on Database Systems, vol. 25, 1-42.

[8] E. Frentzos, K. Gratsias, N. Pelekis, Y. Theodoridis, 2007, “Algorithms for Nearest Neighbor Search on Moving Object Trajec- tories,” Geoinformatica, vol. 11, pp. 159 - 193.

[9] E. Frentzos, K. Gratsias, Y. Theodoridis, 2007, “Towards theNext Generation of Location-based Services.” Proceedings of W2GIS, 2007.

[10] E. Frentzos, K. Gratsias, Y. Theodoridis, 2007, “Index-based Most Similar Trajectory Search.” Proceedings of ICDE, pp. 816-825.

[11] N. Pelekis, Y. Theodoridis, S. Vosinakis, T. Panayiotopoulos, 2006, “Hermes - A Framework for Location-Based Data Management,” Proceedings of EDBT, vol 3896/2006, pp. 1130-1134.

[12] N. Pelekis, E. Frentzos, N. Giatrakos, Y. Theodoridis, 2008, “HERMES: Aggregative LBS via a Trajectory DB Engine,” Proc. SIGMOD,

[13] O. Wolfson, A.P. Sistla, B. Xu, S.J. Zhou, S. Chamberlain, 1999, “DOMINO: databases for moving objects tracking,” Proceedings of the SIGMOD International Conference on Management of Data, pp. 547 - 549.

[14] 김경숙, 임복자, 남광우, 이기준, 2000, “이동객체 컴포넌트 설계 및 구현”, 데이터베이스 연구, 제18권 제 4호, pp. 33-41.

[15] 김경숙, 권오제, 변희영, 조대수, 김태완, 이기준, 2004, “이동객체를 위한 질의처리 컴포넌트의 설계 및 구현”, 개방형 GIS학회 논문지, 제6권 1호, pp 31-50.

[16] 박장유, 남광우, 진희채, 2009, “u-GIS 콘텐츠를 위한 GeoPhoto 콘텐츠 언어의 설계,” 한국공간정보시스템학회 논문지 v.11, no. 1, pp. 35-42.

[17] 임복자, 김경숙, 남광우, 이기준, 2003, “이동객체 컴포넌트 설계 및 구현,” 한국GIS학회 2003년도 공동 춘계 학술대회 논문집, pp. 201-207.

논문접수 : 2011.02.07
수 정 일 : 1차 2011.04.01 / 2차 2011.04.26
심사완료 : 2011.04.27

양 평 우
2007년 군산대학교 컴퓨터정보공학과 이학사
2009~군산대학교 컴퓨터정보공학과 석사과정
관심분야는 데이터베이스, GIS, 데이터스트림

이 용 미
2002년 충북대학교 컴퓨터과학과 이학사
2005년 충북대학교 컴퓨터과학과 이학석사
2005~현재 충북대학교 컴퓨터과학과 박사과정
관심분야는 시공간 데이터베이스, 스트림 데이터 처리, 데이터 마이닝

이 연 식
1982년 전남대학교 전자계산학과 이학사
1984년 전남대학교 전자계산학과 이학석사
1994년 전북대학교 전산응용공학과 공학박사
1997년~1998년 University of Missouri(Kansas City) 교환교수
1986년~현재 군산대학교 컴퓨터정보공학과 교수
관심분야는 객체지향시스템, 능동시스템, 센서 네트워크 에이전트 미들웨어, USN 응용

남 광 우
1995년 충북대학교 전자계산학과 이학사
1997년 충북대학교 전자계산학과 이학석사
2001년 충북대학교 전자계산학과 이학박사
2001년~2004년 한국전자통신연구원 텔레매틱스연구단
2004년~현재 군산대학교 컴퓨터정보공학과 부교수
관심분야는 데이터베이스, GIS, LBS 정책 및 기술, 데이터스트림, 지오센서 네트워크