KIPS Tr. Software and Data Eng.
Vol.6, No.2 pp.85~90 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.2.85

Design of Moving Object Query Processing Based on UDF

Kihyun Yoo[†] · Pyoung Woo Yang[††] · Kwang Woo Nam[†††]

ABSTRACT

Various mobile devices are spreading in recent developments in mobile computing environments. Especially the popularity of mobile devices equipped with GPS has become widespread, and various application services utilizing location information are born. In this paper, we propose a system model for storing and managing the trajectory of moving objects, which is the set of location information of moving objects acquired in continuous time, and the UDF (User-Defined Functions) based trajectory index method which can quickly query the large data set of moving object and the Pre-Materialized table method. Then we compare and evaluate the performance of each method through experiments. Experimental results show that the Pre-Materialized table method is about 1.2 times faster than the UDF based trajectory index method on execution time.

Keywords : Moving Object, Query Processing, Trajectory, UDF

UDF 기반 이동객체 질의 처리 설계 및 구현

유 기 현[†] · 양 평 우[††] · 남 광 우[†††]

요 약

최근 모바일 컴퓨팅 환경의 발달로 다양한 모바일 장비들이 보급되고 있다. 특히 GPS가 탑재된 모바일 장비들의 보급이 활발해지면서 위치 정보를 이용한 다양한 응용서비스들이 생겨나고 있다. 이 논문에서는 연속적인 시간에서 획득한 이동객체 위치 정보들의 집합, 즉 이동객체의 궤적을 저장, 관리하기 위한 시스템 모델 및 대용량 이동객체 데이터를 빠르게 질의할 수 있는 UDF (User-Defined Functions) 기반 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 제안하고 실험을 통해 각 기법들의 성능을 비교 평가한다. 실험에서 질의 선 실체화 테이블 기법이 UDF 기반 궤적 인덱스 기법보다 실행시간에서 약 1.2배 빠른 결과를 보였다.

키워드 : 이동객체, 질의 처리, 궤적, UDF

1. 서 론

최근 모바일 관련 기술의 발달로 스마트폰 및 태블릿 컴퓨터와 같은 다양한 모바일 장치들이 보급되고 있다. 뿐만 아니라, 모바일 컴퓨팅 환경의 발달로 인하여 기존의 기술과 첨단 기술을 융합하여 새로운 서비스를 제공하려는 시도가 계속 되고 있다. 특히 GPS가 탑재된 모바일 장비들의 보급이 활발해지면서 위치 정보를 이용한 다양한 응용서비스들을 제공할 수 있게 되었다. 초창기 위치기반서비스(LBS)에서는 위치 정보에만 관심을 가졌기 때문에 다른 부가 정보들은 크게 주목 받지 못했다. 하지만, 최근 위치 정보에 기반한 서비스들이 과거, 현재, 미래의 상관관계를 고려하여 분석하기 위해 위치 데이터와 시간 데이터를 함께 활용하고 있다. 예를 들면, 실시간으로 도로 교통 현황 및 주변 상황 정보를 제공해주는 실시간 교통 정보 서비스나 이동객체의 GPS좌표를 축적하고, 분석하여 최적의 길 안내 서비스를 제공하는 지능형 네비게이션 서비스, 밤 늦게 귀가하는 청소년 혹은 여성들의 안전한 귀가를 위한 안심귀가 서비스, 통근 시간과 같은 특정 시간대에서 비슷한 경로로 이동하는 사람들을 매칭시켜주는 카풀 서비스 등이 제공되고 있지만, 위치 정보만으로 이러한 서비스들을 신속, 정확하게 제공하는데 한계가 있다. 따라서 과거 데이터 및 연속적인 데이터들을 분석하기 위해 시간 속성이 함께 고려되어야 한다. 궤적은 연속적인 시간에서 획득한 이동객체 위치 정보들의 집합이다. 그리고 이동객체 궤적 데이터의 특성상 단 몇 일만 수집되어도 대량의 데이터셋을 생성한다. 이러한 대용량 궤적 데이터는 일반 데이터베이스에서 처리하기

※ 이 논문은 정부의 재원으로 국토교통부 국토공간정보연구사업의 연구비 지원(14NSIP-B080144-01)과 한국연구재단이 지원하는 일반 연구자 지원 사업(2013R1A1A4A01013416)에 의해 수행된 연구의 결과물임.
† 준 회 원 : 군산대학교 컴퓨터정보통신공학부 박사과정
†† 준 회 원 : 군산대학교 컴퓨터정보통신공학부 석·박사통합과정
††† 종신회원 : 군산대학교 컴퓨터정보통신공학부 교수
Manuscript Received : December 7, 2016
Accepted : December 14, 2016
* Corresponding Author : Kwang Woo Nam(kwnam@kunsan.ac.kr)

가 적합하지 않다. 따라서 이동 객체의 궤적을 저장, 관리하기 위한 시스템 모델 및 대용량 이동객체 데이터를 빠르게 질의할 수 있는 방법에 대한 연구가 필요하다.

대용량 데이터의 처리 및 분석을 위해 일부 응용 환경에서는 DBMS 내에서 처리하지 않고, 전체 데이터셋을 파일로 추출하여 처리한다. 빠른 처리능력이 요구되는 대용량 데이터셋의 경우, 파일로 추출하여 처리하는 방법이 대부분의 경우 DBMS 내에서 SQL 질의를 이용하여 처리하는 것보다 훨씬 빠르기 때문이다. 하지만 DBMS 내에서 파일로 추출할 때 드는 비용과 DBMS에서 지원하는 여러 기능들을 사용할 수 없는 등가교환(Trade Off) 비용을 생각한다면 그 비용을 무시할 수 없기 때문에, DBMS 내에서 대용량 데이터셋을 처리하고자 하는 연구들이 진행되었고 그 방법으로 UDF(User-Defined Functions)를 사용하여 위의 문제들을 해결하고자 했다[1, 2]. UDF는 사용자 정의 함수를 뜻하며, 사용자가 SQL을 이용하여 만들거나 C 언어와 같은 C-레벨 코드로 오브젝트 파일을 생성하고, 생성된 오브젝트 파일을 직접 DBMS 내부에 설치하여 만들어진 함수들을 말한다. 다시 말하면, 사용자가 DBMS 소스 코드를 건드리지 않고, DBMS를 확장할 수 있는 API(Application Programming Interface)로써 동작하는 방법이라고 할 수 있다. 이러한 UDF 함수는 DBMS 내에서 표준 SQL 명령어와 같이 사용할 수 있다.

최근 궤적 데이터에 대한 연구가 활발해지면서 몇몇 DBMS들이 미비하게나마 궤적 데이터를 처리할 수 있는 함수들을 지원하고 있다. 하지만, 아직까지 궤적 데이데에 대해 완벽히 지원하는 시스템은 존재하지 않는다. 따라서 이 논문에서는 일반 DBMS에서 궤적 데이터를 처리하기 위해 UDF 기반으로 확장 가능한 궤적 데이터 저장, 관리 시스템 모델을 소개한다. 또한, 대용량 궤적 데이터를 빠르게 질의할 수 있는 UDF 기반 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 제안하고 실험을 통해 각 기법들의 성능을 비교 평가한다.

이 논문의 구성은 다음과 같다. 2장에서는 이동객체와 궤적 데이터의 처리에 관한 관련 연구를 소개하고, 3장에서는 궤적 데이터를 처리하기 위한 UDF 기반 궤적 시스템 모델과 질의 예제들을 설명한다. 4장에서는 궤적관계 질의 처리 전략에 대해 설명하고, 5장에서 각각의 질의 처리 방법에 대한 성능분석을 보인다. 마지막으로 6장에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

DOMINO(Database for MovINg Objects)[3, 4]는 DBMS에서 이동객체를 지원하기 위해 이동객체 시공간 모델(MOST)을 제안하였다. 이 MOST 모델은 이동객체의 현재 위치를 획득하기 위해서 방향 벡터를 저장하고 있다. 또한, 가까운 미래 위치를 예측할 수 있는 FTL(Future Temporal Logic) 질의 언어를 제안하였다. 오라클에서 시공간 질의를 지원할 수 있도록, 오라클 DBMS의 확장 형태로 개발된 HERMES[5, 6]는 ORDBMS 에 시공간 질의를 가능하게 하는 연산자들을 제공한다. HERMES의 주요 목표는 연속적인 이동객체의 모델링과 질의의 지원이다. HERMES는 데이터 타입의 집합과 그에 상응하는 연산자들을 정의하였다. HERMES의 ORDBMS 층에는 오라클 ORDBMS 서버에 궤적 데이터 저장과 LBS 지원을 위한 내부 구조의 질의 능력에 적합하게 강화되었다. Güting et al.[7-10]은 ORDBMS에 과거와 현재 데이터 모두 저장하기 위한 데이터 모델을 제안하였다. 또한, 모든 세그먼트들을 무한 개의 커브 셋으로 표현할 수 있는 추상 데이터 타입과 유한 개의 폴리라인으로 표현할 수 있는 불연속 데이터 타입을 제안하였다. SECONDO[11, 12]는 공간 객체를 확장하여 이동객체 데이터 타입과 연산자들을 정의하였고, 이 연산자들을 구현하기 위한 알고리즘을 제안하고 있다. 또한, SECONDO는 이동객체 데이터베이스를 위한 벤치마크 시스템인 BerlinMOD[13]를 지원한다.

앞서 소개한 연구들은 일반 DBMS에서 궤적 데이터를 저장, 관리할 수 있는 모델링 방법들에 초점을 맞추고 있다. 하지만 궤적 데이터는 일정한 주기로 연속적으로 수집되기 때문에 일부 응용에서는 대용량의 데이터셋을 만들기도 한다. 일반 DBMS에서는 검색해야 될 데이터셋이 커질수록 검색 성능이 크게 저하된다. 또한, 대용량 데이터셋을 검색하기 위해 많은 시스템 자원을 소비해야 한다. 따라서 대용량의 궤적 데이터셋을 효율적으로 저장, 관리할 수 있는 시스템 모델 및 질의 방법들의 연구가 필요하다. 이 논문에서는 대용량 이동객체 궤적 데이터의 저장 및 관리를 위한 모델링 방법을 설명하고, 궤적 데이터의 질의 성능을 향상시키기 위한 UDF 기반의 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 제안한다.

다음에서는 이 논문에서 제안하고 있는 UDF 기반의 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 소개하고 실험을 통하여 각 기법들의 대한 성능 평가를 보인다.

3. 시스템 모델

3.1 UDF 기반 궤적 시스템 모델

궤적은 연속적인 시간 흐름에 따라 바뀌는 이동객체 위치 정보들의 셋이다. 때문에 궤적 데이터를 표현할 때 일반적으로 특정 시간 (t)에서의 위치 좌표 (x, y)를 (x, y, t)의 셋으로 표현한다. 궤적 데이터는 데이터의 연속적인 특성 때문에, 모든 궤적 데이터들이 시간 순서에 따라 계속해서 저장소에 삽입된다. 이렇게 삽입되는 데이터들은 짧은 시간동안 많은 양의 데이터 셋을 수집하게 된다. 따라서 대용량 궤적 데이터를 빠르게 검색해줄 수 있는 색인 기법이 요구된다. 이 논문에서는 이러한 대용량 궤적 데이터의 색인 기법으로 UDF 기반의 3D R-tree 인덱스를 생성하는 방법과 질의 성능을 향상시키기 위하여 임시 뷰 테이블을 생성하고 정제 과정을 수행하는 질의 선 실체화 전략을 제안한다.

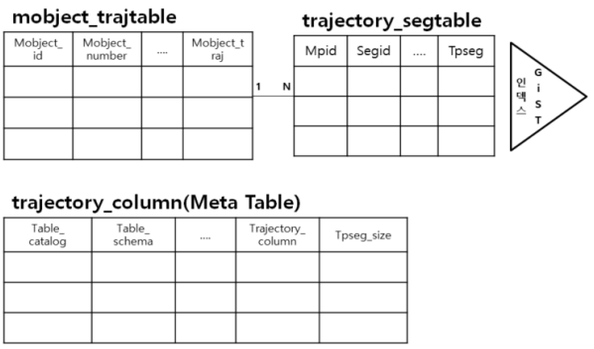

Fig. 1. Table Scheme and Meta Table

Fig. 1은 궤적 데이터를 저장하기 위한 궤적 테이블(mobject_trajtable)과 순차적으로 생성되는 궤적 세그먼트들을 저장하는 궤적 세그먼트 테이블(trajectory_segtable), 그리고 각 궤적들의 메타 정보를 저장할 수 있는 메타 테이블(trajectory_column)을 나타내고 있다.

mobject_trajectory : 사용자가 궤적 정보에 대하여 질의할 수 있는 사용자 정의 테이블이다. 실제로 사용자는 이 사용자 정의 테이블만을 이용하여 궤적 정보에 대하여 질의할 수 있다.

trajectory_segtable : 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 구별할 수 있도록 하였다. 하나의 segid에는 addTrajectory Column에서 지정해준 숫자(tpseg의 size)만큼의 위치 정보를 저장하게 된다. rect는 한 segid에 있는 위치들을 둘러싼 rectangle을 표현하기 위한 정보이다. 또한 next_segid와 before_segid를 만들어 각 segid를 double linked list처럼 관리를 할 수 있다.

trajectory_column : 사용자가 지정한 trajectory_column의 이름과 자동 생성된 trajectory_segtable의 이름 등 각 테이블에 대한 전체적인 정보를 갖고 있는 메타 테이블이다. 해당 객체의 고유 moid와 객체의 trajectory를 저장할 테이블의 oid를 갖는다.

궤적 데이터를 저장하기 위한 가장 간단한 방법은 한 개의 위치 좌표와 시간 값을 한 로우에 저장하는 것이다. 하지만, 이러한 방법으로 데이터를 저장할 경우에 몇 가지 문제를 야기할 수 있다. 첫째, 주기적으로 계속 생성되는 궤적 데이터를 한 로우에 하나씩 삽입하게 되면 저장되는 데이터의 용량을 불필요하게 증가시킨다. 둘째, 궤적 데이터에 대하여 질의할 때 질의 결과로 반환되는 데이터들의 셋이 커지므로 검색 성능에 상당한 영향을 끼칠 수 있다. 즉, 성능 저하를 야기시킬 수 있다.

이 논문에서는 궤적 데이터를 저장할 때 지정한 개수의 궤적 세그먼트 셋으로 저장한다. 궤적 세그먼트 셋은 이동 객체가 이동한 점 포인트와 그 순간에서의 시간으로 구성된 Tpoint (Time Point) 타입의 Array 형태로 만들어진다. Tpoint는 지오메트리 타입의 점 객체(Point)와 시간(Timestamp) 속성으로 구성된다.

3.2 질의 예

다음은 3.1절에서 소개하고 있는 궤적 시스템 모델을 위한 테이블 생성 질의 및 궤적 컬럼을 생성하는 방법과 궤적 데이터에 대한 사용자 질의를 나타내고 있다.

```
CREATE TABLE taxi
( taxi_id int, taxi_number char(20),
  taxi_model char(20), taxi_driver char(20) );

SELECT addtrajectorycolumn
('public', 'taxi', 'traj', 4326, 'MOVINGPOINT', 2, 150);
```

위의 테이블 생성 질의는 이동객체의 메타 정보를 저장할 수 있는 Object Table을 생성한다. 그리고 다음 질의는 생성된 Object Table에 궤적 컬럼을 추가하고, 실제 궤적 데이터를 저장, 관리하는 Trajectory Segment Table 을 생성한다. 이때, 함수의 매개변수로는 순서대로 데이터베이스의 scheme, 궤적 컬럼을 추가할 Object Table의 이름, 궤적 컬럼의 이름, 공간 데이터의 SRID, 데이터 타입, 데이터 타입의 차원, 한 로우에 저장되는 궤적 세그먼트의 개수를 가리킨다.

```
SELECT taxi_id, taxi_number, TJ_Traj(traj)
FROM taxi
WHERE TJ_Passes(traj, TJ_BOX(116.35, 39.93, 116.22, 40.14,
        PERIODS('2008-02-02 13:30:44', '2008-02-02 15:54:46')));
```

위 사용자 질의는 특정 공간 영역을 지정한 시간 주기 동안 통과한 이동객체들의 ID와 Number, 그리고 각 이동객체의 궤적 정보를 가져온다. 위 질의에서 TJ_Traj(), TJ_Passes(), TJ_BOX() 함수들이 이 논문에서 제안하고 있는 UDF 기반의 확장 함수들이다. TJ_Traj() 함수는 이동 객체의 궤적 세그먼트 셋인 Tpoint Array를 반환하고, TJ_Passes() 함수는 특정 공간 영역과 지정한 시간 주기 동안 통과한 이동객체들의 유/무를 검사한다. TJ_BOX() 함수는 box2d 형태의 공간 바운더리 (x min, y min, x max, y max)와 시간 주기 PERIODS(시작 시간, 종료 시간)를 함수의 매개변수로 입력받아 시공간 질의를 가능하게 하는 UDF 함수이다.

```
SELECT count(*)
FROM taxi
WHERE TJ_Inside(traj, TJ_BOX(116.35, 39.93, 116.22, 40.14,
        PERIODS('2008-02-02 13:30:44', '2008-02-02 15:54:46')));
```

위 질의는 특정 공간 영역 내부에 지정한 시간 주기 동안 검색된 이동객체들의 개수를 가져오는 질의이다. 이전 질의와 마찬가지로 TJ_BOX() 함수를 이용하여 시공간 질의를 수행한다.

```
SELECT taxi_id, taxi_number
FROM taxi
WHERE TJ_Cross(traj, taxi_id, TJ_BOX(116.35, 39.93, 116.22, 40.14,
        PERIODS('2008-02-02 13:30:44', '2008-02-02 15:54:46')));
```

위 질의는 특정 공간 영역과 지정한 시간 주기 동안 특정 이동객체와 만나거나 교차했던 이동객체들의 ID와 Number 정보를 가져온다. 이와 같이 UDF 함수를 이용하면 질의 처리 함수들을 새로 정의하거나 재정의 함으로써 사용자가 원하는 형태로 질의 결과를 가져올 수 있을 뿐만 아니라, 질의 결과에 대한 최적화 또한 가능하다.

4. 궤적관계 UDF 질의처리 전략

4.1 단순 궤적 UDF 기법(Naive)

단순 궤적 UDF(User-Defined Functions) 질의는 궤적 질의의 조건부에서 질의 조건을 만족하는 레코드의 유/무를 검사한다. 이 논문에서는 단순 궤적 UDF 질의를 위해 2차원 공간 데이터의 위상관계 함수들을 확장하여 시공간 궤적 데이터의 위상관계를 표현할 수 있는 함수들로 구현하였다. 특정 공간을 통과한 궤적들의 유/무를 검사하는 TJ_Passes() 함수와 특정 공간 안에 머물고 있는 궤적들의 유/무를 검사하는 TJ_Inside() 함수가 궤적의 위상관계를 표현하기 위해 확장하여 사용하고 있는 함수들이다.

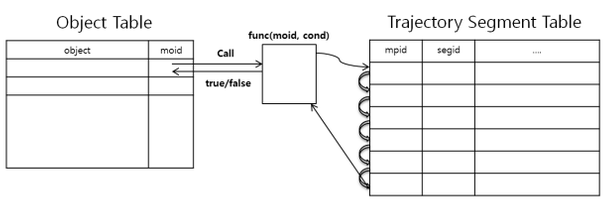

Fig. 2. Simple Trajectory UDF Query Processing

3장에서 설명하고 있는 궤적 시스템 모델에서 이동객체에 대한 질의는 이동객체의 메타 정보를 저장하고 있는 Object Table과 이동객체의 궤적 정보를 저장하고 있는 Trajectory Segment Table과의 조인으로 이루어진다. 이때, 사용자는 직접 두 개의 테이블을 조인하지 않고 사용자에게 제공되는 궤적 UDF 함수를 호출하여 질의한다. UDF함수에서는 Object Table의 moid와 질의 조건(cond)이 일치하는 레코드를 검색하기 위해 Trajectory Segment Table을 스캔하고, 일치하는 레코드의 존재 유/무에 따라 true/false 값을 반환한다. Fig. 2는 단순 궤적 UDF의 질의 처리 과정을 나타내고 있다. 단순 궤적 UDF 질의는 전체 레코드 셋을 스캔하기 때문에 질의 처리에 드는 비용이 많이 발생한다. 이 논문에서는 이러한 질의 처리 비용을 줄이기 위한 UDF 기반 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 다음 단락에서 소개하고 있다.

4.2 UDF 기반 궤적 인덱스 기법(GiST-R-NT)

궤적 데이터는 기본적으로 2차원의 공간 데이터와 1차원 시간 데이터가 결합된 3차원 데이터라고 할 수 있다. 이 논문에서는 이러한 궤적 데이터를 위한 인덱스 알고리즘으로서 GiST(Generalized Search Tree) 인덱스를 확장한 3D R-tree 인덱스를 구현하였고, 구현된 3D R-tree 인덱스를 사용하여 질의 처리 시간을 비교하였다. GiST는 B-tree와 R-tree 또는 그 변형 트리들을 쉽게 구현할 수 있도록 지원하는 템플릿 트리이다. 즉, R-tree 인덱스를 구현하기 위해서 엔지니어는 최소 6개의 핵심 함수들만을 고려하여 구현할 수 있게 되었다. 질의 조건을 검사하는 Consistent(E,q) 함수와 입력 엔트리들을 병합해 주는 Union(P) 함수, 삽입되는 엔트리의 노드 위치를 결정하기 위해 필요한 Penalty (E1,E2) 함수, 노드의 분할을 결정해 주는 PickSplit(P) 함수, 입력 엔트리를 저장 가능한 포맷으로 변경해주는 Compress (E) 함수, Compress(E) 함수의 역함수인 Decompress(E) 함수들이 제공되고 있다.

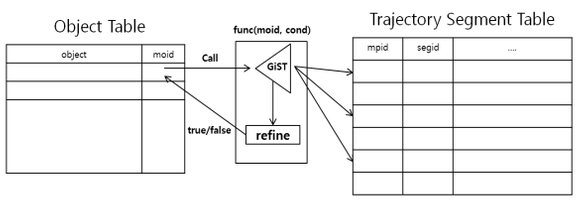

Fig. 3. UDF Based Trajectory Index Query Processing

UDF 기반 궤적 인덱스 질의는 이전 장에서 소개한 단순 궤적 UDF 질의에서 사용한 궤적 위상 함수를 사용한다. 질의 과정은 Trajectory Segment Table로부터 생성한 GiST 인덱스 데이터셋에서 질의 조건을 만족하는 후보 레코드셋을 검색하고, 정제(refine) 단계를 거쳐서 실제 레코드의 유/무를 검사한다. 그리고 단순 궤적 UDF 질의와 같이 true/false 값을 반환한다. Fig. 3은 위에서 설명하고 있는 UDF 기반 궤적 인덱스 질의 처리 과정을 나타낸다.

PostgreSQL/PostGIS에서는 GiST기반의 2차원 R-tree를 지원할 뿐만 아니라, n차원 R-tree로 확장할 수 있는 함수들을 따로 제공하고 있다. 이 논문에서는 PostgreSQL/PostGIS에서 지원하는 GiST 인덱스의 n차원 R-tree 지원함수들을 확장하여 3D R-tree를 구현하였다.

4.3 질의 선 실체화 테이블 기법(GiST-R-MT)

대용량 데이터의 검색 성능을 향상시키기 위해서 일반적으로는 검색하고자 하는 데이터 셋에 인덱스를 생성하는 방법을 사용한다. 이 논문에서는 검색 성능을 향상시키기 위한 다른 방법으로 질의 선 실체화 테이블 기법을 이용하고 있다.

질의 선 실체화 테이블 기법은 Materialized 뷰로 알려져 있는 기법으로, 처음 질의 요구가 있을 때 임시 뷰 테이블을 물리적으로 생성하고 뷰에 다른 질의들이 이어질 것이라는 가정 하에 생성된 임시 뷰 테이블을 유지하는 방법이다. 이때 뷰의 내용을 최신 상태로 유지하기 위해서 기본 테이블이 갱신되었을 때 뷰 테이블을 자동으로 갱신하는 효율적인 방법이 고려되어야 한다. 점진적 갱신의 개념을 이용한 방식은 이런 목적으로 개발되었는데, 이 방법은 정의된 기본 테이블들 중 하나에 변경 사항이 적용될 때 실체화된 뷰 테이블 내에 삽입, 삭제, 갱신해야 할 튜플들을 결정한다. 일반적으로 이 뷰는 질의가 있는 동안 유지된다. 일정 기간 동안 뷰에 질의가 없으면 시스템은 물리적인 뷰 테이블을

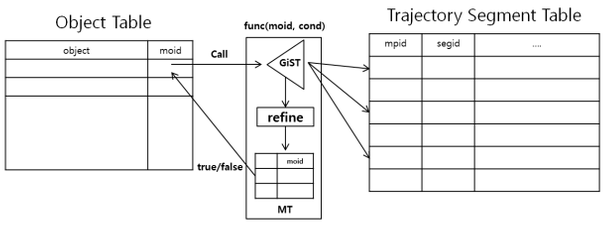

Fig. 4. Pre-Materialized Table Query Processing

자동적으로 삭제하고, 다음에 그 뷰를 참조하는 질의가 있을 때 처음부터 뷰 테이블을 재구성한다.

Fig. 4는 질의 선 실체화 전략의 질의 처리 과정을 보이고 있다. 먼저 사용자 질의가 있을 때, 질의 조건을 만족하는 Trajectory Segment Table의 mpid 들로 임시 테이블(MT)을 생성한다. 그리고 UDF 기반 궤적 인덱스 질의와 같이 GiST 인덱스 데이터셋에서 질의 조건을 만족하는 후보 레코드셋을 정제시킨 레코드들의 mpid가 생성된 임시 테이블에 존재하는지를 검사한 후 검사 결과에 따라 true/false 값을 반환한다. 이렇게 질의 선 실체화 전략을 사용했을 때 기존 질의 방법보다 실행 시간에서 약 1.2배의 성능 향상을 보였다.

5. 구현 및 실험

5.1 실험 환경

이 절에서는 UDF 기반 궤적 인덱스로써, 3DR-tree를 사용하여 질의 처리 성능에 대한 실험을 수행한다. 또한, 이 논문에서 질의 성능을 향상시키기 위해 제안하고 있는 질의 선 실체화 테이블 기법을 사용하여 각 기법들의 성능에 대하여 비교, 분석한다. 실험은 인텔 코어 i5 3.2GHz CPU와 8GB 메모리를 가진 컴퓨터에서 수행하였고, 운영체제로는 리눅스 우분투 14.04 버전을 사용하였다. 실험에 사용된 데이터는 실제 중국 베이징시의 택시들로부터 수집된 GPS 궤적 데이터로서, 택시들로부터 수집된 대량의 GPS 데이터를 이용하여 신속하고 정확한 그리고 빠르고 안전한 경로로 길 안내를 하기 위한 서비스를 만들기 위하여 수집되었다. 이 데이터는 약 3만대의 택시로부터 6개월가량 수집된 데이터이다.

Table 1. Experimental Parameter

Parameter	Setting
Page size	4k
Node capacity	50, 100, **150,** 200, 250
Query window size	2%, 4%, **6%,** 8%, 10%
Number of queries	200
Dataset size	100K, 300k, **500K,** 700k, 1M

5.2 성능평가

Table 1은 질의 성능을 측정하는 실험에서 사용된 파라미터들이다. 실험은 4장에서 설명하고 있는 3개의 질의 방법들에 대하여 사용자 궤적 질의를 수행하여 걸린 시간을 측정하였다.

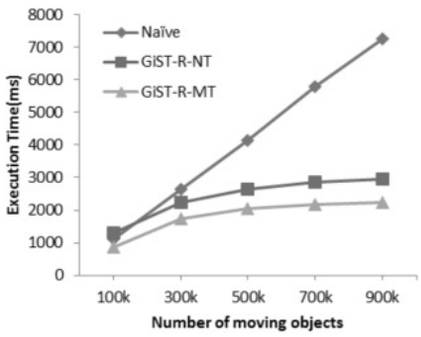

Fig. 5. Execution Time per Data Set Size

Fig. 5는 데이터셋의 크기가 증가할 때 각각의 질의 실행 시간을 측정하고 있다. 실험에서 단순 궤적 UDF 질의(Naive)는 데이터 셋이 증가하면서 질의 실행 시간이 선형적으로 증가하는데 반해, UDF 기반 궤적 인덱스 질의 기법(GiST-R-NT)과 질의 선 실체화 기법(GiST-R-MT)은 데이터셋이 증가하면서 질의 실행 시간이 완만하게 증가되는 것을 볼 수 있다. 여기서 주목할 점은 질의 선 실체화 기법이 UDF 기반 궤적 인덱스 질의보다 실행 시간에서 약 1.2배의 성능 향상을 보였다는 것이다.

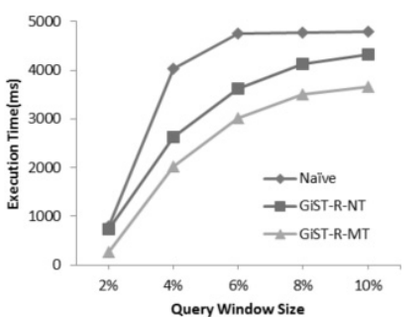

Fig. 6. Execution Time per Query Window Size

Fig. 6은 질의 윈도우 크기에 따른 질의 실행 시간을 측정하였다. 이 실험에서 3개의 질의들 모두 질의 윈도우 크기가 증가할 때, 전체적으로 완만한 실행 시간을 보인다. 질의 윈도우 크기가 2%에서 4%로 증가할 때 실행 시간 변동 폭이 가장 크게 나타나는데, 이것은 윈도우 크기가 2%일 때 검색되는 데이터 로우의 개수가 너무 적어서 상대적으로 실행 시간 또한 작게 측정되었기 때문이다.

6. 결 론

최근 모바일 컴퓨팅 환경의 발달로 다양한 모바일 장비들이 보급되고 있다. 특히 GPS가 탑재된 모바일 장비들의 보급이 활발해지면서 위치 정보를 이용한 다양한 응용서비스들이 생겨나고 있다. 시간에 따른 이동객체의 위치 변화를 이용한 이동객체 궤적에 대한 연구들이 이러한 영향으로 활발하게 진행되고 있다. 궤적은 연속적인 시간에서 획득한 이동객체 위치 정보들의 집합이다. 그리고 이동객체 궤적 데이터의 특

성상 단 몇 일만 수집되어도 대량의 데이터셋을 생성한다. 이러한 대용량 궤적 데이터는 일반 데이터베이스에서 처리하기가 적합하지 않다. 따라서 이동 객체의 궤적을 저장, 관리하기 위한 시스템 모델 및 대용량 이동객체 데이터를 빠르게 질의할 수 있는 방법에 대한 연구가 필요하다. 이 논문에서는 이동객체의 궤적을 저장, 관리하며 대용량 이동객체 데이터를 빠르게 질의 할 수 있는 UDF 기반 궤적 인덱스 기법과 질의 선 실체화 테이블 기법을 제안하고 실험을 통해 각 기법들의 성능을 비교 평가한다.

향후 대용량 궤적 데이터의 처리비용을 줄이기 위한 방법으로 대용량 궤적 데이터의 압축 기법들에 대한 연구가 필요하다.

References

[1] Z. Chen and C. Ordonez, "Efficient OLAP with UDFs," in *Proceedings of ACM 11th International Workshop on Data Warehousing and OLAP*, pp.41-48, 2008.

[2] C. Ordonez and C. Garcia-Alvarado, "A data mining system based on SQL queries and UDFs for relational databases," in *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, pp.2521-2524, 2011.

[3] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving objects databases: Issues and solutions," in *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pp.111-22, 1998.

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

[5] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos, "Hermes - A Framework for Location-Based Data Management," in *Proceedings of EDBT*, pp.1130-1134, 2006.

[6] N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis, "HERMES: Aggregative LBS via a Trajectory DB Engine," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.1255-1258, 2008.

[7] M. Erwing, R. H. Güting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Type: An Approach to Modeling and Querying Moving Object in Databases," *GeoInformatica,* Vol.3, No.3, pp.269-296, 1999.

[8] L. Forlizz, R. H. Güting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Object Databases," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp.319-330, 2000.

[9] R. H. Güting, M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," *ACM Transactions on Database Systems*, Vol.25, No.1, pp.1-42, 2000.

[10] J. A. C. Lema, L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider, "Algorithms for moving objects databases," *The Computer Journal,* Vol.46, No.6, pp.680-712, 2003.

[11] S. Dieker and R. H. Güting, "Plug and play with query algebras: SECONDO - A generic DBMS development environment," in *Proceedings of the International Database Engineering and Applications Symposium*, pp.380-392, 2000.

[12] R. H. Güting, T. Behr, V. T. de Almeida, D. Ansorge, Z. Ding, T. Höse, F. Hoffmann, and M. Spiekermann, "SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching," in *Proceedings of the 21st International Conference on Data Engineering,* pp.1115-1116, 2005.

[13] C. Duntgen, T. Behr, and R. H. Güting, "BerlinMOD: A Benchmark for Moving Object Databases," *The VLDB Journal,* Vol.18, No.6, pp.1335-1368, 2009.

유 기 현

e-mail : khyoo1221@kunsan.ac.kr
2007년 군산대학교 컴퓨터정보공학과 (이학사)
2009년 군산대학교 컴퓨터정보공학과 (이학석사)
2014년~현 재 군산대학교 컴퓨터정보통신공학부 박사과정
관심분야 : 데이터베이스, GIS, 데이터스트림

양 평 우

e-mail : manner7979@kunsan.ac.kr
2007년 군산대학교 컴퓨터정보공학과 (이학사)
2009년~현 재 군산대학교 컴퓨터정보통신공학부 석・박사통합과정
관심분야 : 데이터베이스, GIS, 데이터스트림

남 광 우

e-mail : kwnam@kunsan.ac.kr
1995년 충북대학교 전자계산학과(이학사)
1997년 충북대학교 전자계산학과(이학석사)
2001년 충북대학교 전자계산학과(이학박사)
2001년~2004년 한국전자통신연구원 텔레매틱스연구단
2004년~현 재 군산대학교 컴퓨터정보통신공학부 교수
관심분야 : 데이터베이스, GIS, 데이터스트림, 지오센서 네트워크