Supporting Trajectory UDF Queries and Indexes on PostGIS

Kwang Woo NAM and Pyoung Woo YANG

Kunsan National University and Turbosoft Inc.

This research was supported by a Grant (14NSIP-B080144-01) from National Land Space Information Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

Introduction

- **Moving Objects and Trajectories**
  - GPS Everywhere!

Phone

Car

Bike

and, life log for human

images from
Apple
https://www.wired.com/2009/06/black-box-for-the-car/
http://www.bayareabikeshare.com

Introduction

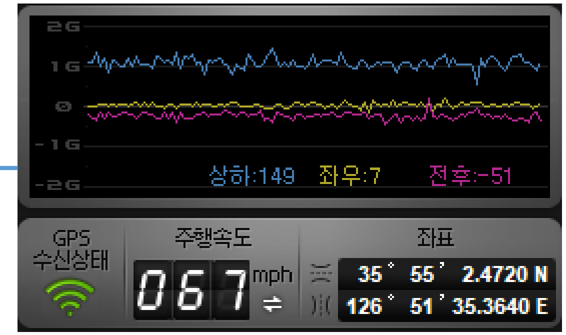
- **In modern cities, many people drive the vehicles that equipped with the GPS devices**
  - It is easily being collected and stored the GPS data

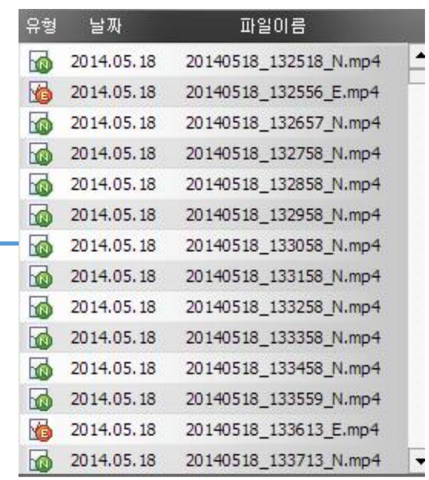
- **Many applications for location-based services(LBS) and moving object have been studied using these GPS data**

- **Moreover, we can use a large-scale GPS data**
  - Because it is easily collected from the vehicles
  - Recently it is increasing studies for mining a meaningful and a valuable information from the large-scale data

Introduction

- **Real Example : Trajectories in Car Blackbox**
  - Moving Point and Moving Double



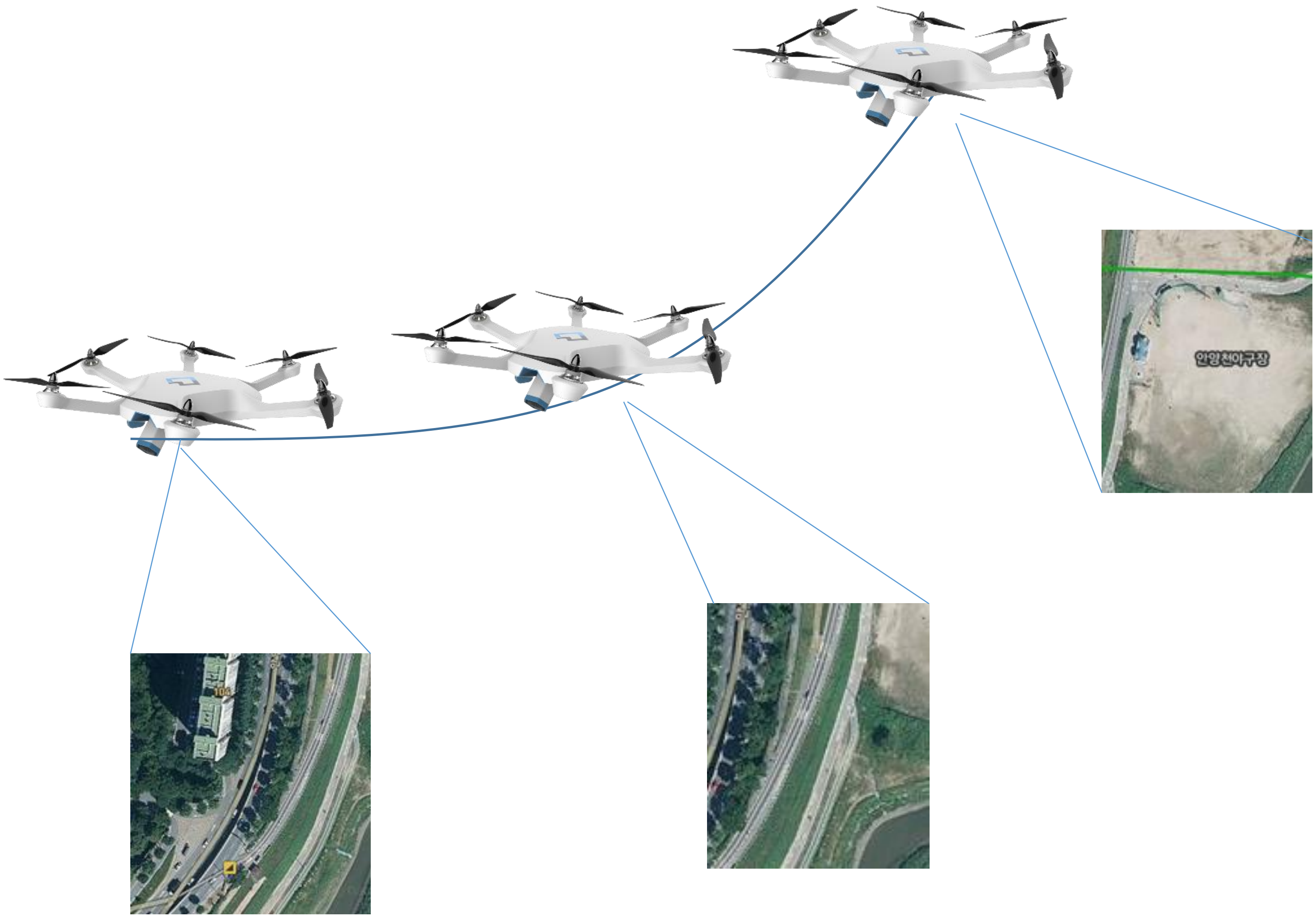


유형	날짜	파일이름
N	2014.05.18	20140518_132518_N.mp4
E	2014.05.18	20140518_132556_E.mp4
N	2014.05.18	20140518_132657_N.mp4
N	2014.05.18	20140518_132758_N.mp4
N	2014.05.18	20140518_132858_N.mp4
N	2014.05.18	20140518_132958_N.mp4
N	2014.05.18	20140518_133058_N.mp4
N	2014.05.18	20140518_133158_N.mp4
N	2014.05.18	20140518_133258_N.mp4
N	2014.05.18	20140518_133358_N.mp4
N	2014.05.18	20140518_133458_N.mp4
N	2014.05.18	20140518_133559_N.mp4
E	2014.05.18	20140518_133613_E.mp4
N	2014.05.18	20140518_133713_N.mp4

*.mp4 : video data
*.gps : gps data(NMEA)
*.3gf : acceleration sensor

Introduction

- **Real Example : Trajectories in Drone**

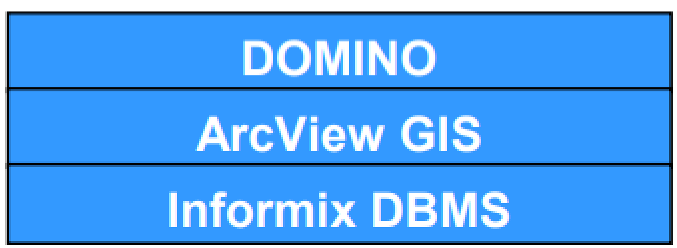


Introduction

- **The trajectory is the set of information of the location by the time**
  - Unfortunately traditional spatial database systems do not support data types and functions for trajectory data
- **PostgreSQL is probably one of the best solutions for trajectory data**
  - It is an open-source ORDBMS(Object-Relational DBMS)
  - Supports objects, classes and inheritance in database schemas and query language
- **Large-scale trajectory data is useful**
  - By analyzing and predicting the trajectory data, it provide a new opportunity to understand the city dynamics and economic phenomena

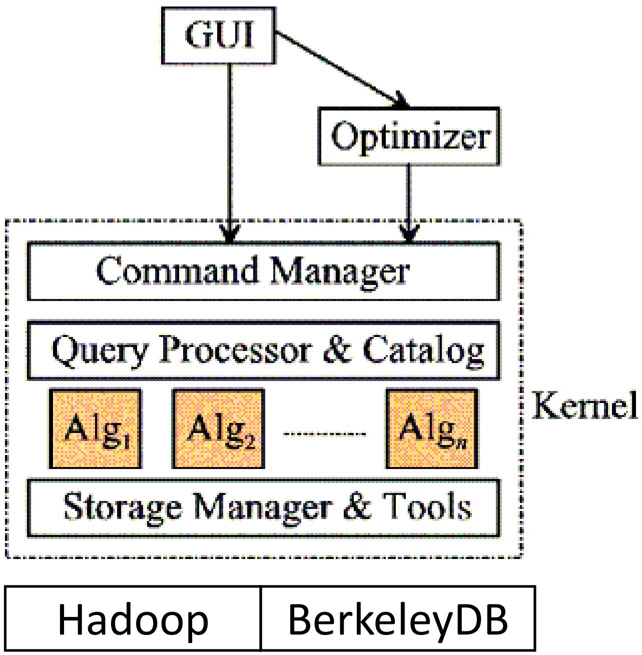
Related Work

- **DOMINO**
  - DOMINO(Database fOr MovINg Objects)
    - Professor Ouri Wolfson
    - University of Illinois at Chicago
  - Model and Language
    - Moving Objects Spatio-Temporal
    - FTL Query Language for Spatio-temporal Query



Related Work

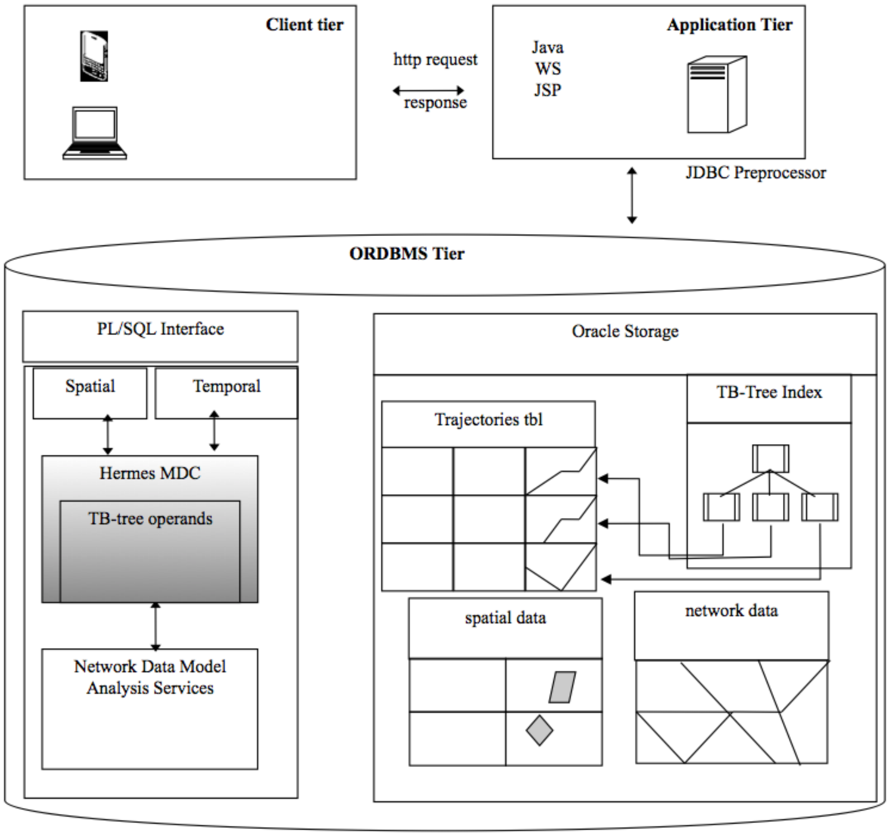
- **SECONDO**
  - FernUniversitat, Germany
    - Professor Ralf Harmut Guting
    - http://dna.fernuni-hagen.de/secondo/




```
Secondo => let SeqQuery = fun()
Secondo ->   Trips feed addcounter[No, 1]
Secondo ->   extend[Matched:
Secondo ->     omapmatchmht(Edges, EdgeIndex_Box_rtree, EdgeIndex,
.Trip) aconsume]
Secondo ->   extend[MTraj: .Matched afeed projecttransformstream[Cu
rve] collect_line[TRUE]]
Secondo ->   consume feed
Secondo -> project[MTraj] consume;
```

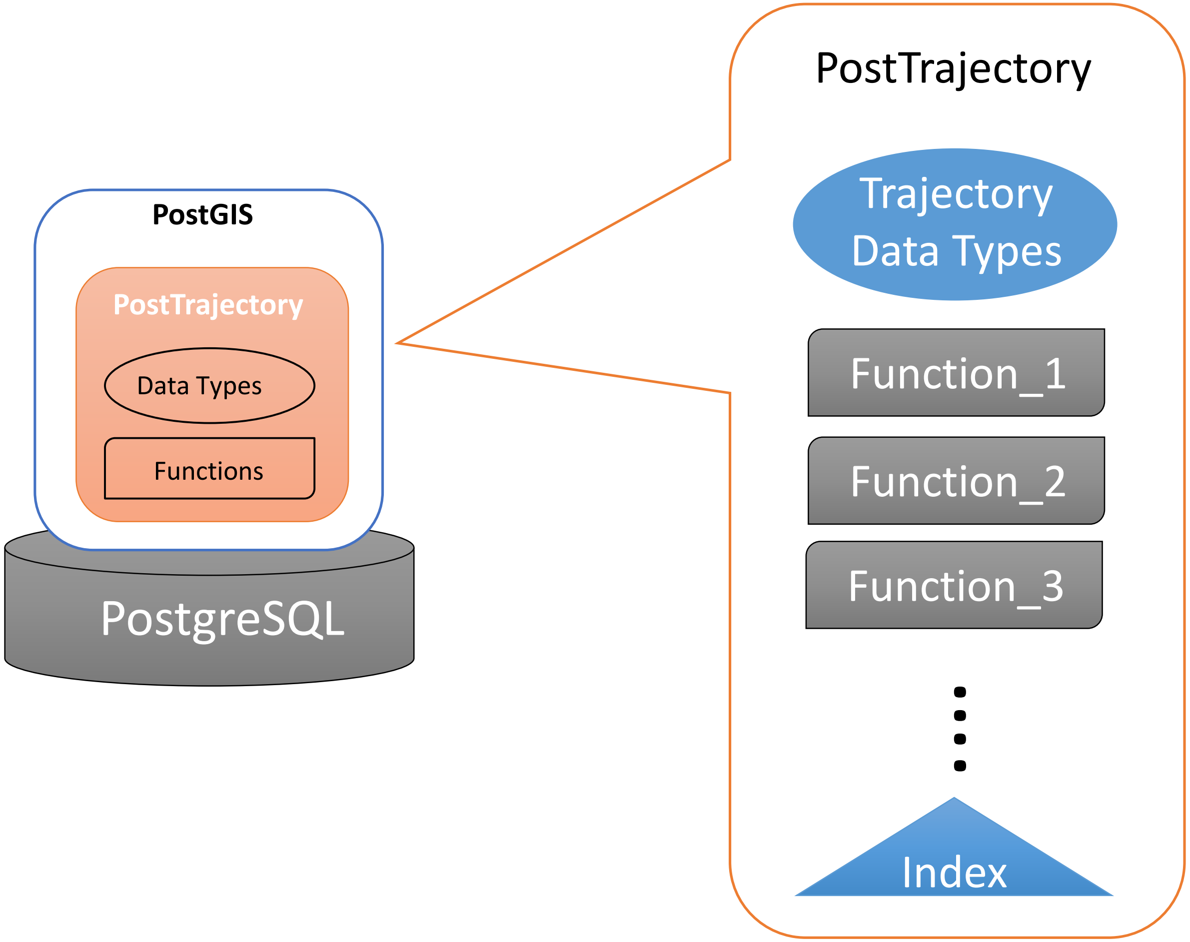
Related Work

- **HERMES on ORACLE**
  - University of Piraeus
    - Professor Yannis Theodoridis



System Overview

- **Architecture**

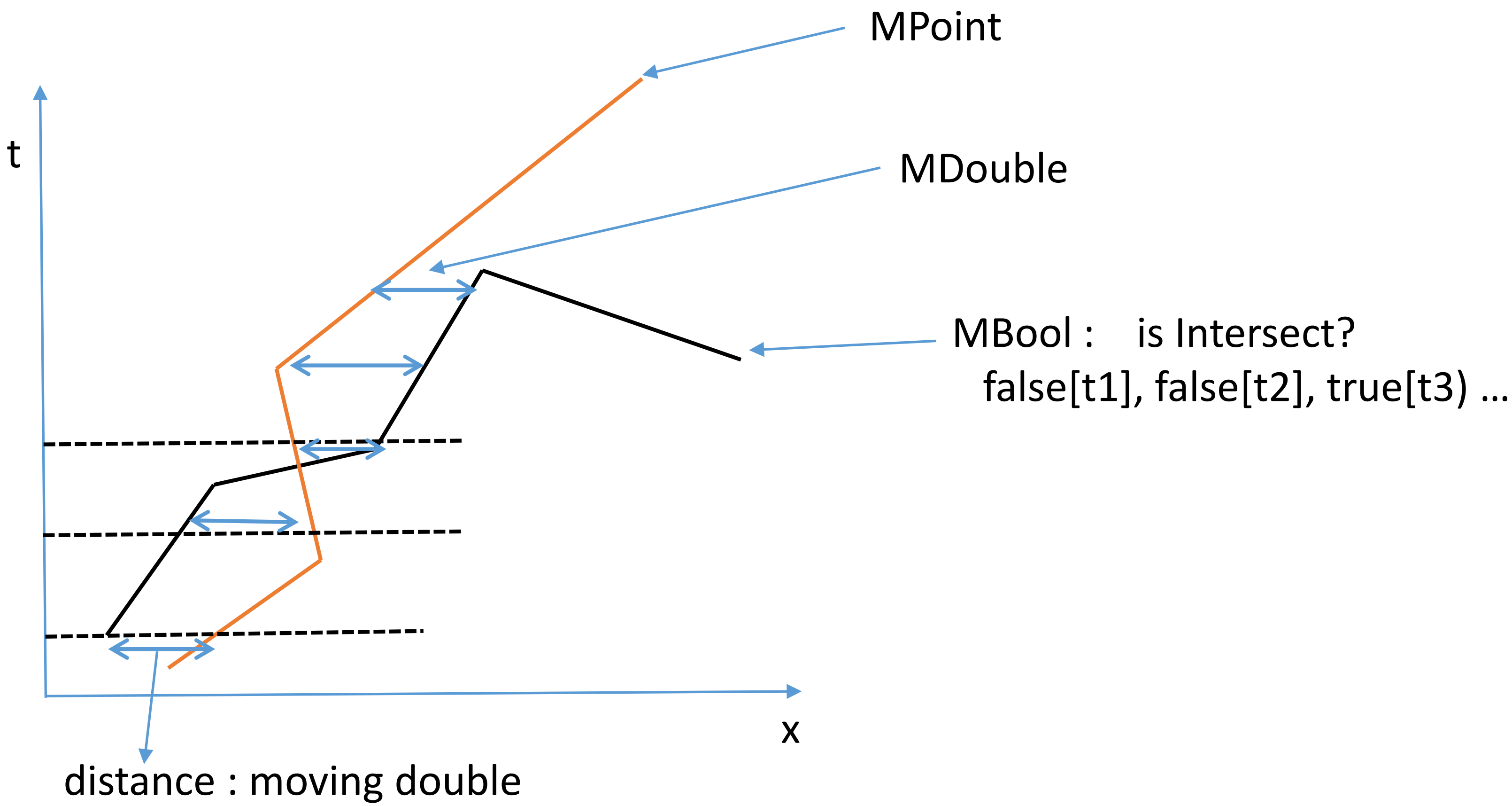


Data Model

- **Data Type**
  - TPoint and TDouble
    - TPoint : (Point, Timestamp)
    - TDouble : (Double, Timestamp)
    - TBool : (Boolean, Timestamp)
  - MPoint
    - TPoint[]
  - MDouble
    - TDouble[]
  - MBool
    - TBool[]

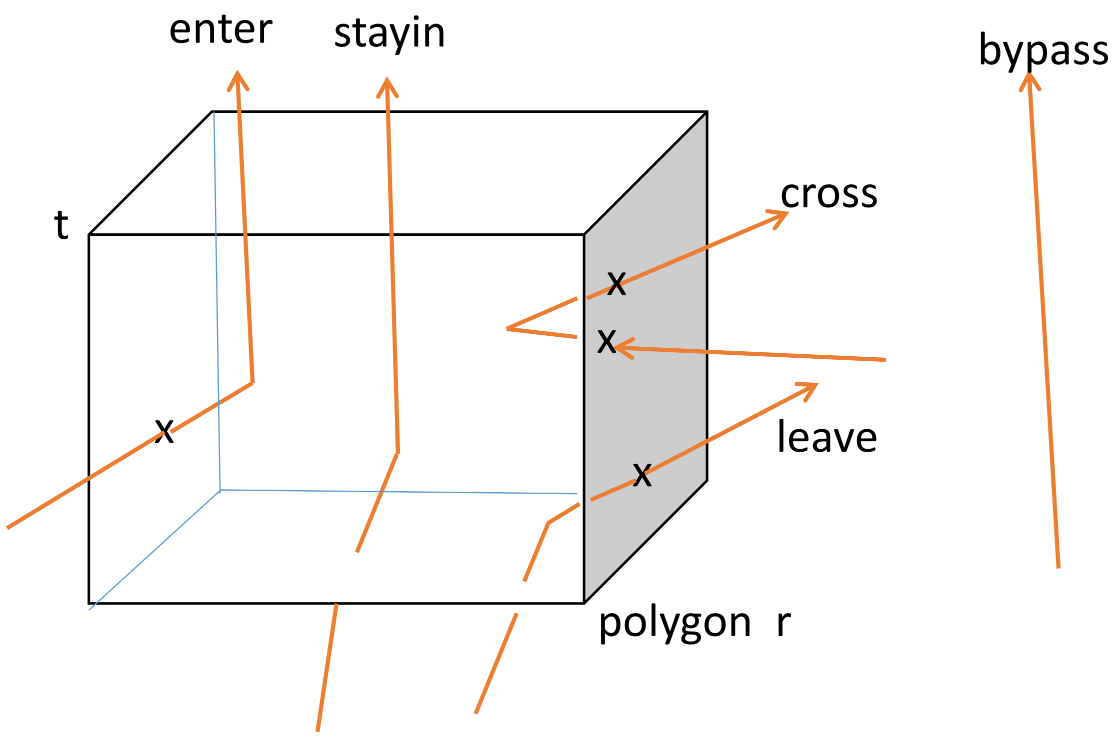
Data Model

- **Data Type and Operations**



Data Model

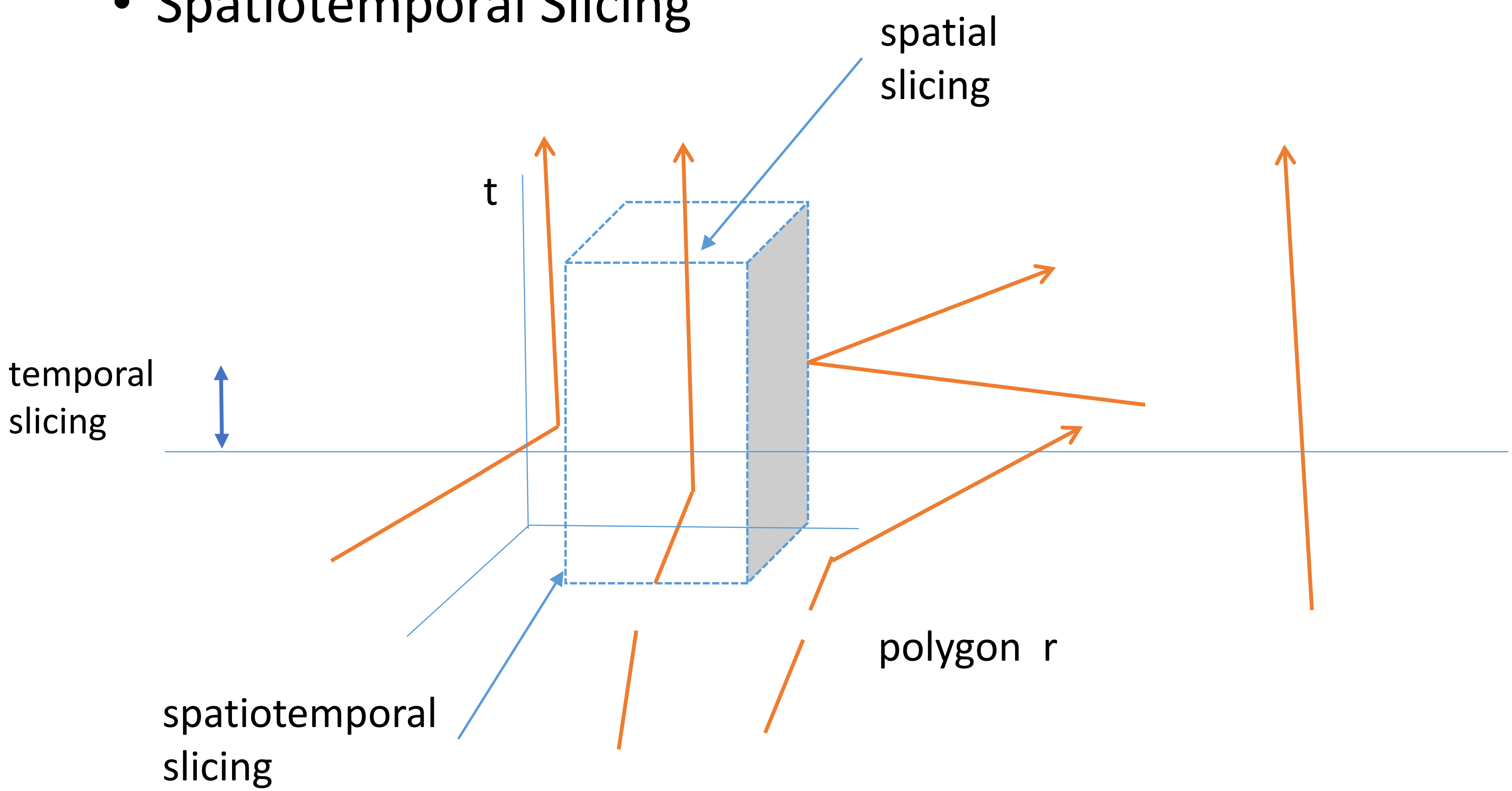
- **Spatiotemporal Relationship Operations**
  - Enters
  - Leaves
  - Crosses
  - StayIn
  - Bypass



Data Model

- **Slice Operations**
  - Temporal Slicing
  - Spatial Slicing
  - Spatiotemporal Slicing

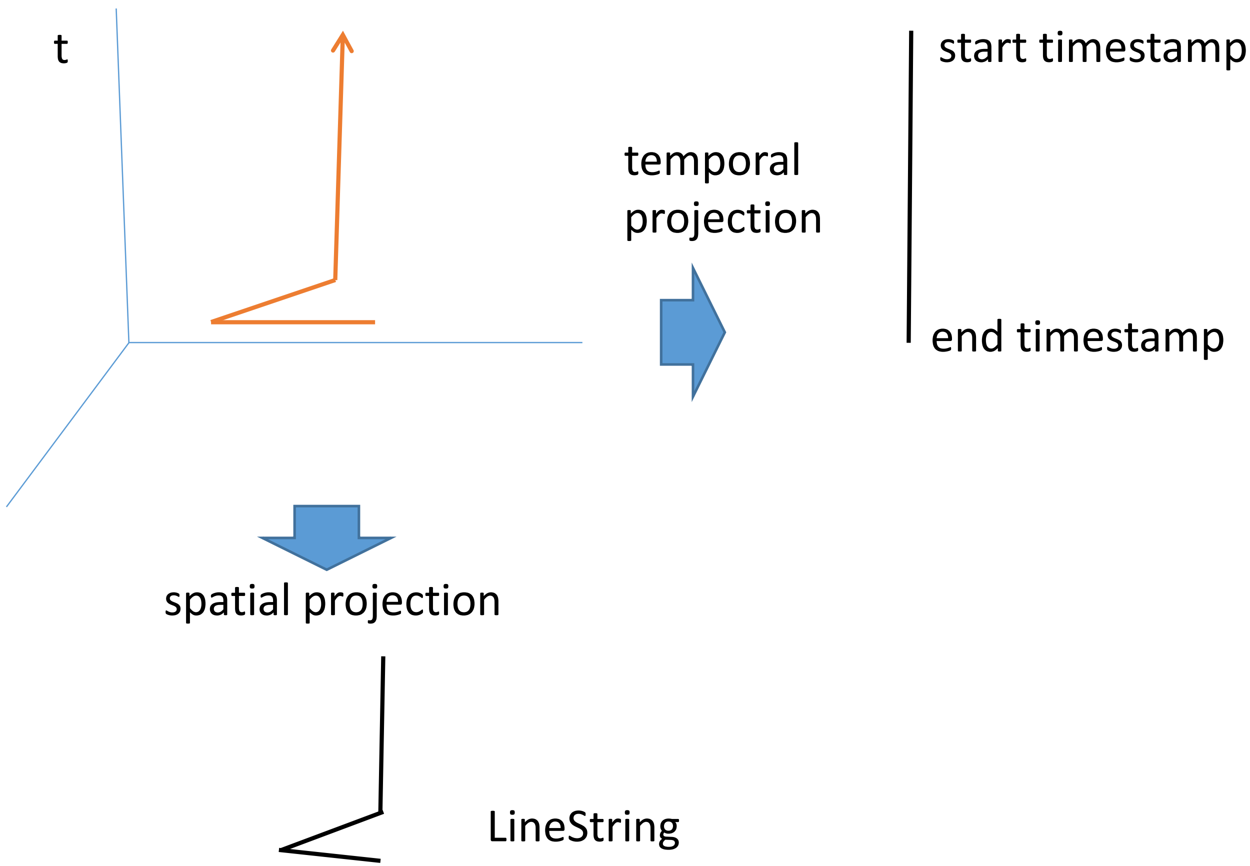
mpoint ←slice(mpoint, period)
mpoint ←slice(mpoint, geometry)
mpoint ←slice(mpoint, geometry, period)



Data Model

- **Projection Operations**
  - Spatial Projection
  - Temporal Projection

LineString ←sproject(mpoint)
period←tproject(mpoint)



Storage Model

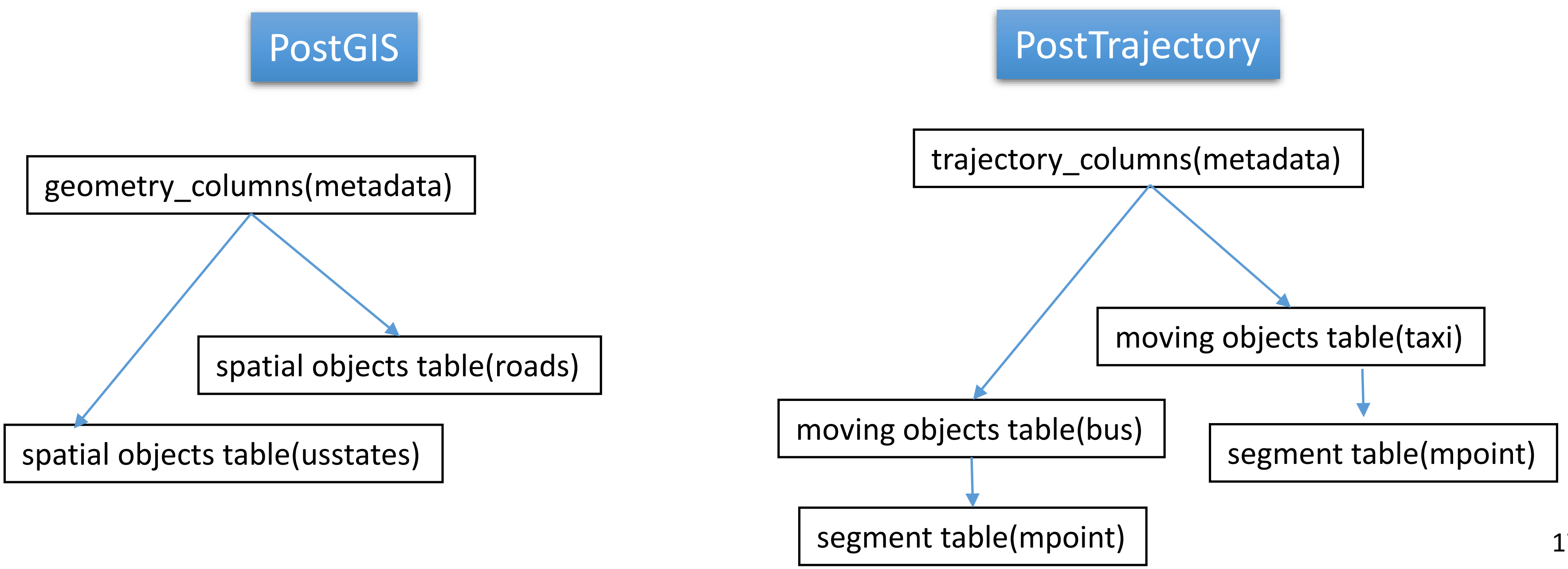
- **Table Creation**

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

SELECT AddTrajectoryColumn('public', 'taxi', 'traj', 4326, ‘MPOINT', 2, 150);
```

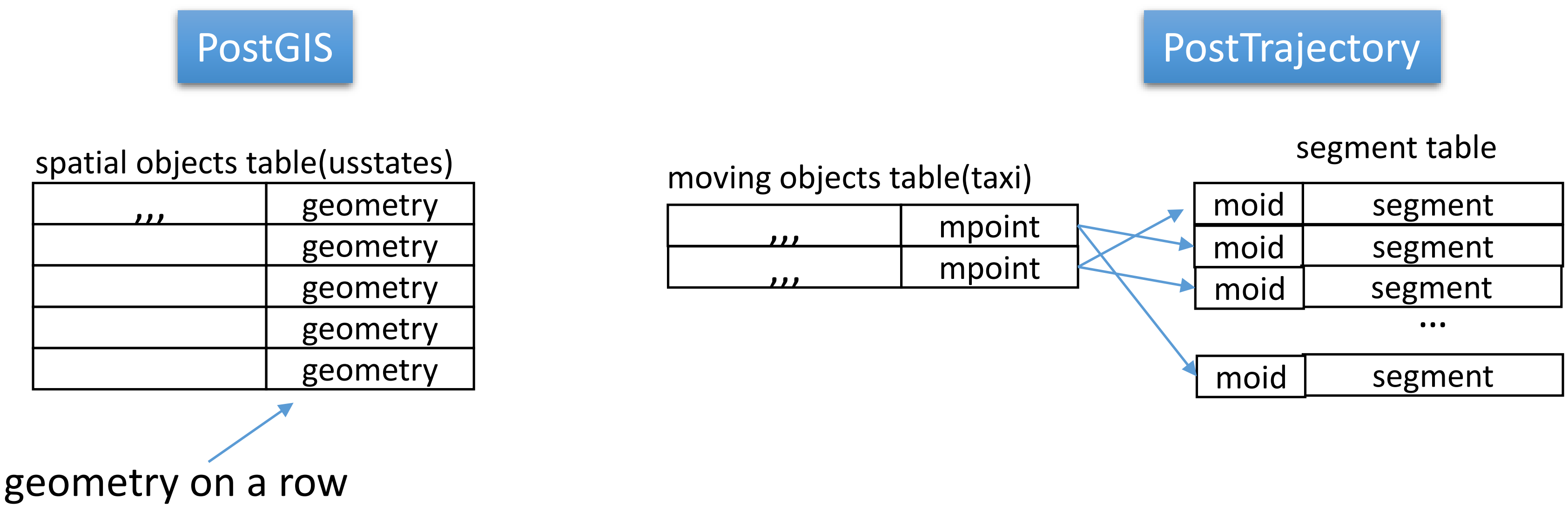
Storage Model

- **Tables**
  - Meta Data Table : Trajectory_Column
  - Moving Objects Table
  - Segment Table
    - A trajectory is split into segments



Storage Model

- **MPoint**
  - (segtable_oid, moid)
  - One segment table per a trajectory attribute



mpoint := (segtable_oid, moid)

Storage Model

- **Example**

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

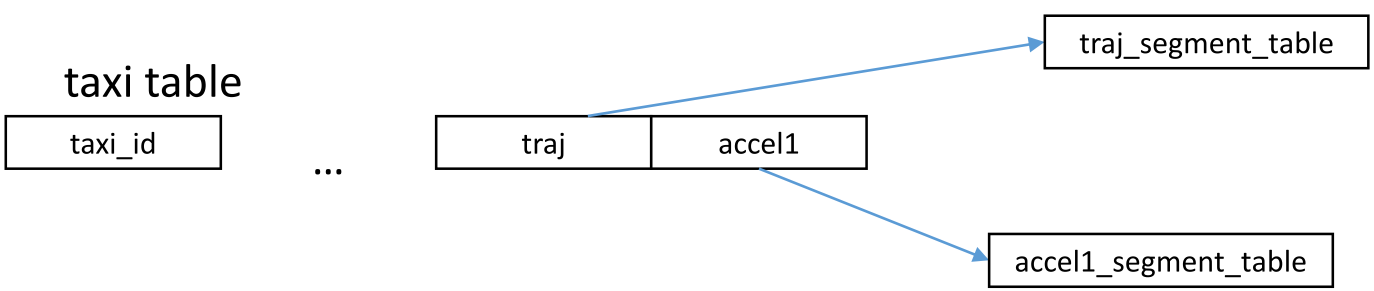
georeference WGS4326

2 Dminsion

150 tpoint per a segment

```
SELECT AddTrajectoryColumn('public', 'taxi', 'traj', 4326, ‘MPOINT', 2, 150);
SELECT AddTrajectoryColumn('public', 'taxi', ‘accel1', 4326, ‘MDouble', 2, 150);
```

taxi table



Storage Model

- **Segment Table**

moid	segid	tpcount	rect	start_time	end_time	tpseg

box2d

timestamp

tpoint[]

- strategies
    - split
        - count-based-split(*)
        - spatial-based-split
        - temporal-based-split
        - st-based-split
    - compression
        - no compression : tpoint[]
        - naïve compression : zip
        - simplification

Trajectory Queries

- **Insert new Moving Object**

```
##Inserting Moving Objects
insert into taxi values(1, '57NU2001', 'Optima', 'hongkd7');
insert into taxi values(2, '57NU2002', 'SonataYF', 'hongkd7');
```

- **Append GPS Trajectory for a moving object**

```
UPDATE taxi
SET     traj = append(traj, tpoint(st_point(200, 200),
                                   timestamp '2010-01-25 12:05:30+09'))
WHERE  taxi_id = 1;
```

- **Remove GPS Trajectories**

```
UPDATE taxi
SET traj = remove(traj, to_timestamp(12345678), to_timestamp(23456789) )
WHERE taxi_id = 1;
```

Trajectory Queries

- **Temporal Slicing**

```
SELECT tj_slice( traj,
                 timestamp '2010-01-26 14:50:40+09',
                 timestamp '2010-01-26 15:20:40+09')
FROM taxi;
```

- **Spatial Slicing**

```
SELECT tj_slice(traj,
          geometry(‘polygon ( ( 300 200, 300 300, 440 300, 440 200, 300 200 ) )'))
FROM taxi;
```

Trajectory Queries

- **Composite Query with Slicing**

```
SELECT tj_slice( traj,
                 timestamp '2010-01-26 14:50:40+09',
                timestamp '2010-01-26 15:20:40+09')
FROM taxi
WHERE tj_overlap( traj, tj_period( to_timestamp(2432432343), to_timestamp(2432433000));
```

```
SELECT tj_slice( traj, timestamp '2010-01-26 14:50:40+09', timestamp '2010-01-26 15:20:40+09')
FROM taxi
WHERE
  tj_overlap( tj_slice(traj, geometry(‘polygon ( ( 300 200, 300 300, 440 300, 440 200, 300 200 ) )')),
          tj_period(timestamp '2010-01-26 15:00:00+09', timestamp '2010-01-27 00:00:00+09'));
```

Trajectory Queries

- **Spatiotemporal Predicate**

```
SELECT count(*)
FROM taxi
WHERE tj_enter(traj, geometry(‘polygon ( ( 300 200, 300 300, 440 300, 440 200, 300 200 ) )'))
```

- **Spatiotemporal Predicate with Slicing**

```
SELECT taxi_id, tj_slice(traj, geometry(‘polygon( ( 300 200, 300 300, 440 300, 440 200, 300 200 ) )‘))
FROM  taxi
WHERE tj_enter(traj, geometry(‘polygon ( ( 300 200, 300 300, 440 300, 440 200, 300 200 ) )'))
```

Trajectory Queries

- **Simple Distance Queries**

```
SELECT taxi_id, tj_distance(traj, geometry('Point( 50 50 )' ),
                tj_mindistance(traj, geometry('Point( 50 50 )' ),
                tj_maxdistance(traj, geometry('Point( 50 50 )' )
FROM taxi;
```

- **Distance in WHERE**

```
SELECT taxi_id, taxi_number
FROM taxi
WHERE tj_getDistance( tj_mindistance(traj,  geometry('point ( 50 50 )') ) ) < 500;
```

Trajectory Queries

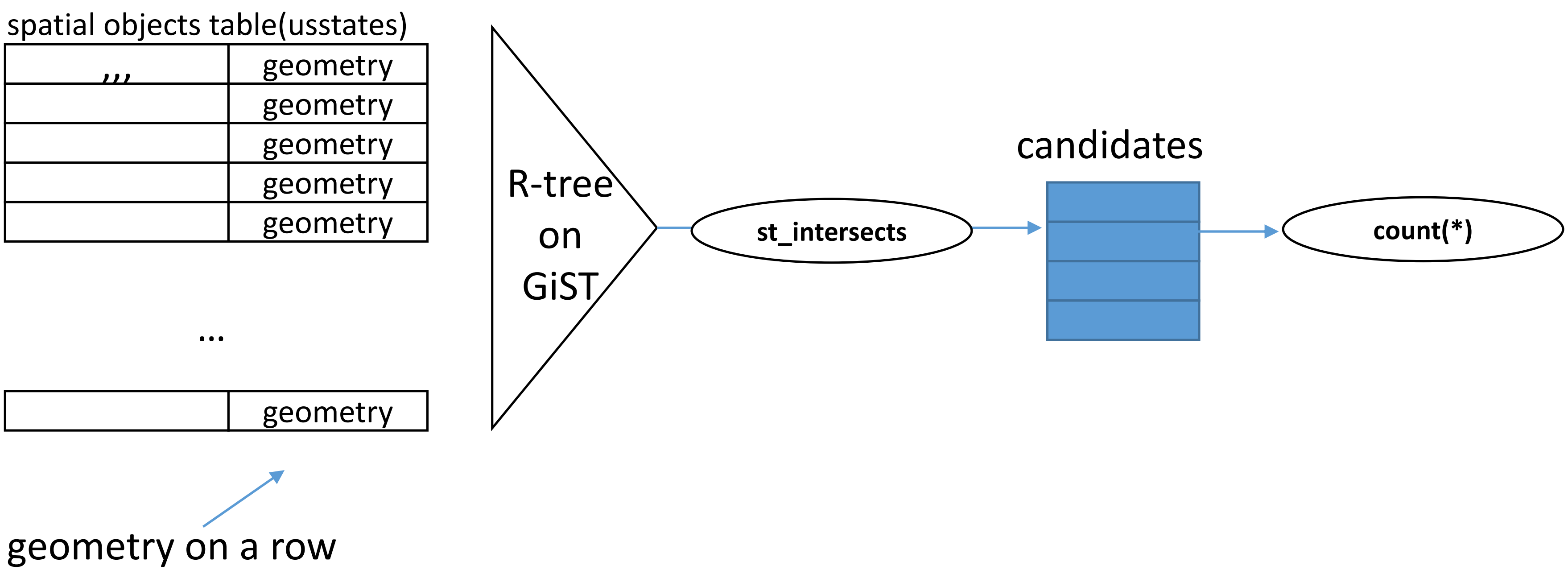
- **Join Distance**

```
SELECT taxi_id, bus_id, tj_distance( t.traj, b.traj)
FROM taxi t, bus b;
```

Supporting Index on Trajectory

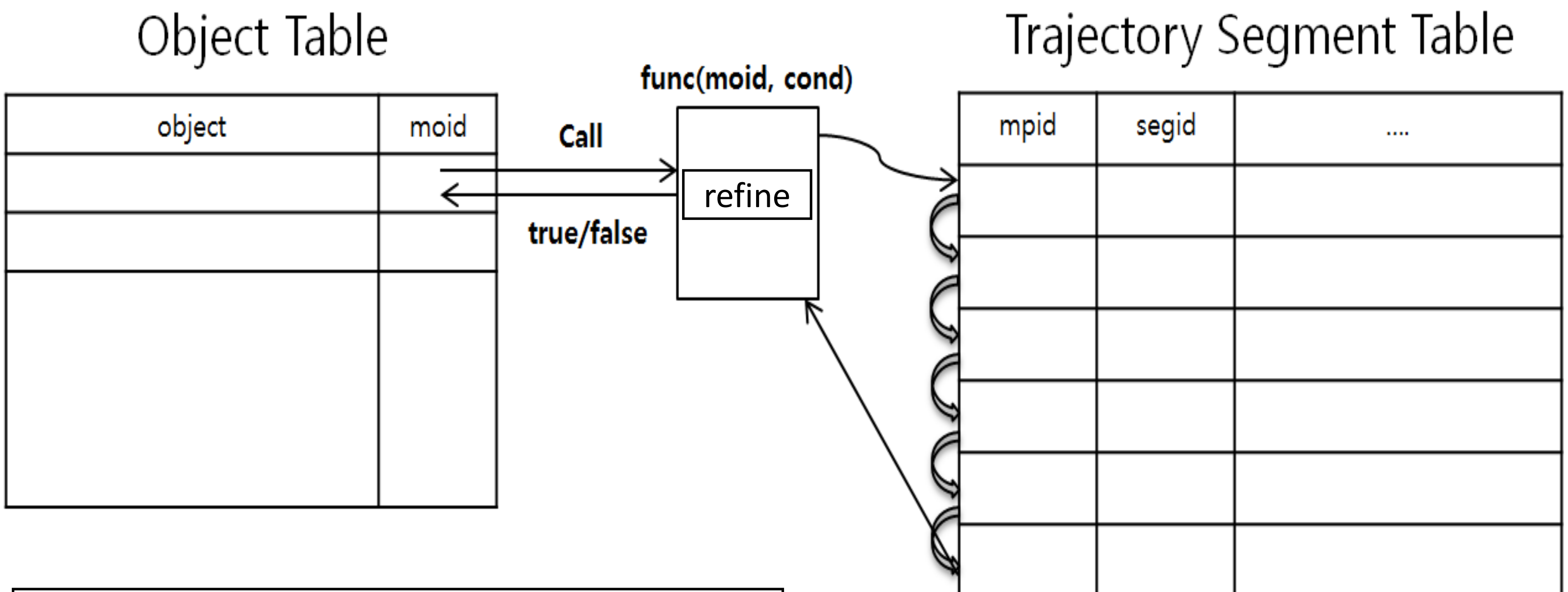
- **PostGIS**
  - R-tree on GiST

```
SELECT   count(*)
FROM     roads
WHERE    st_intersect( geom, $1 )
```



Supporting Trajectory Indexes

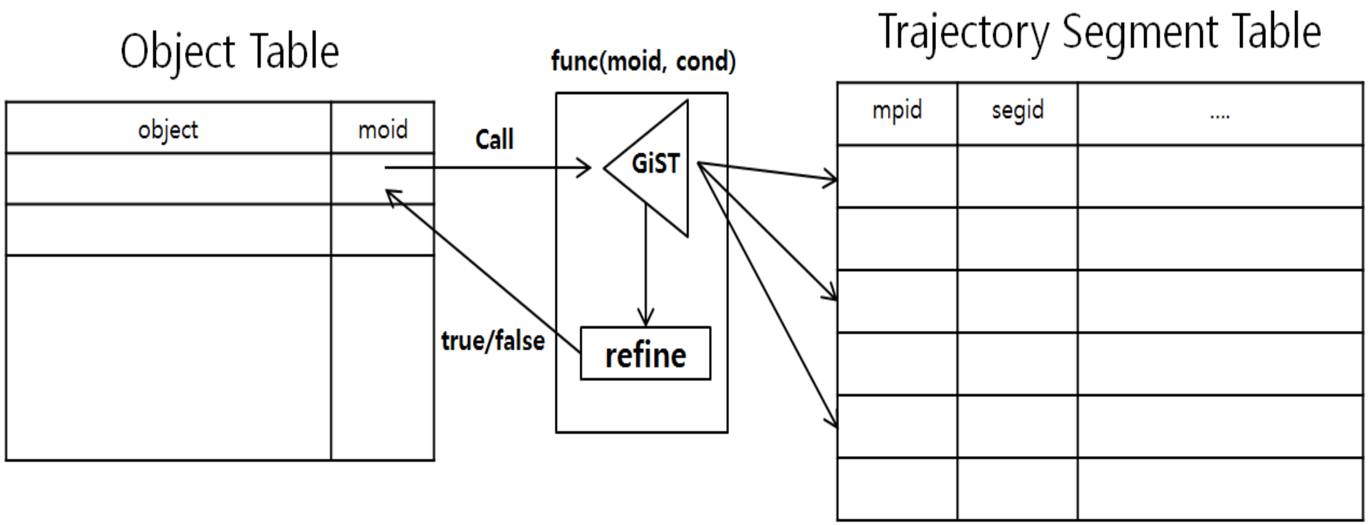
- **Naïve Approach : No support**
  - Separated Segment Table
    - function will be executed a row by row
    - filtering : only use rect attribute on segtable




```
SELECT   count(*)
FROM     taxi
WHERE    tj_crosses( traj, $1 )
```

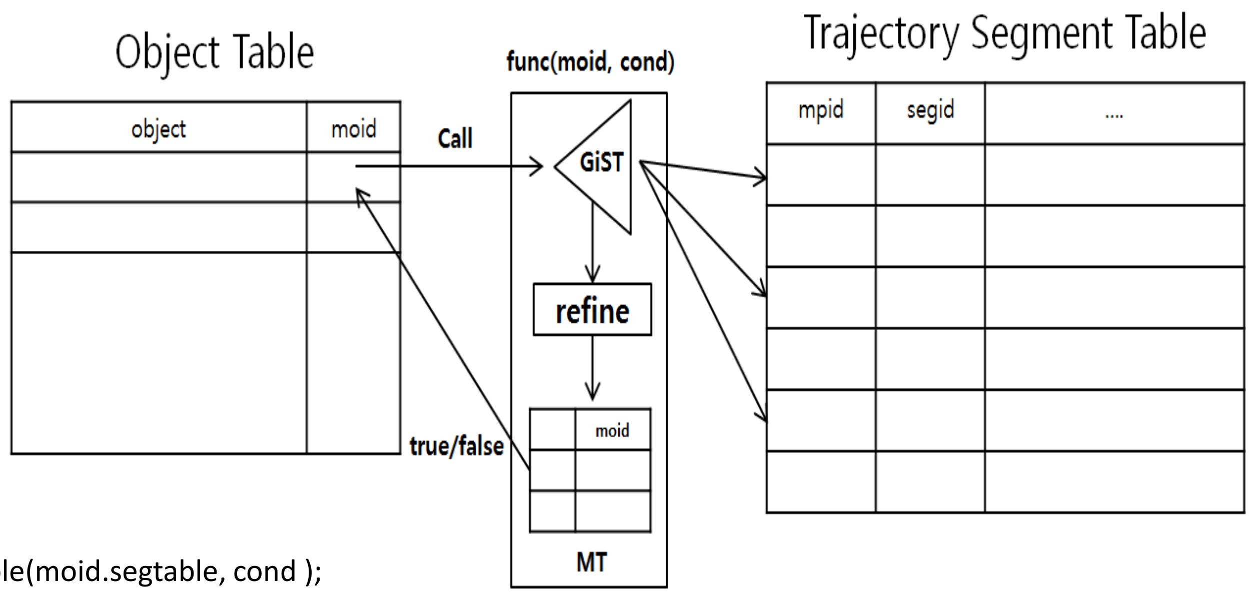
Supporting Trajectory Indexes

- **GiST Extension Approach**
  - GiST are extended for temporal predicate
    - function will be executed a row by row
    - filtering : GiST index for trajectory



Supporting Trajectory Indexes

- **Query Materialization Approach**
  - GiST are extended for temporal function
  - Materialized Table
    - Filtering and Refinement will be just 1-time at first row



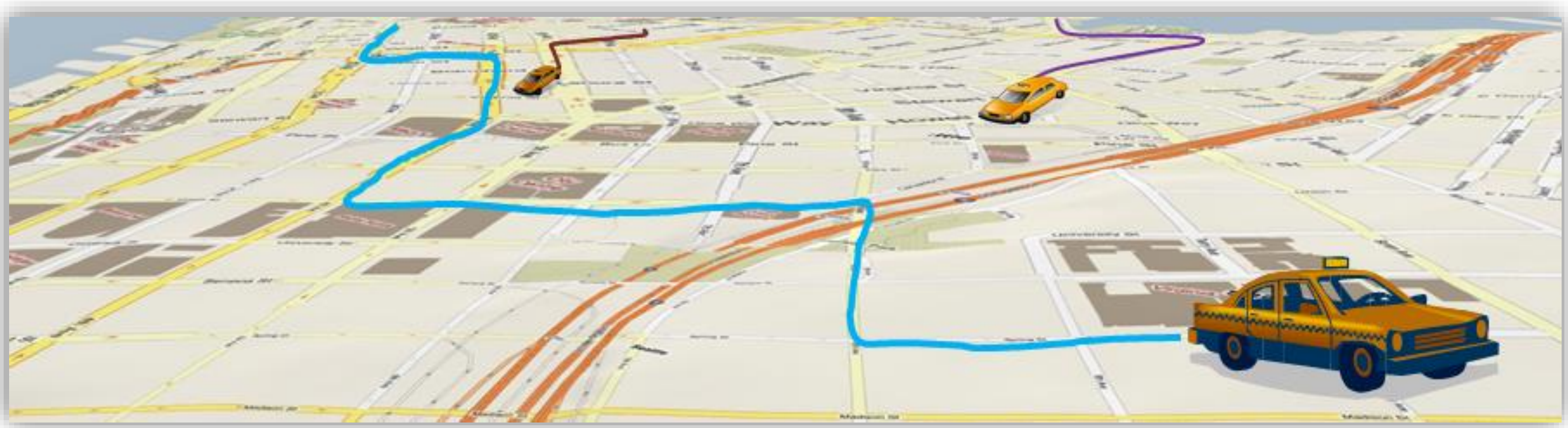

```
function( moid, cond )
{
    if  isFirstExec( cond )
    {
        m = makeMaterizationTable(moid.segtable, cond );
    }

    return lookup( m, moid.mpid );
}
```

Performance Evaluation

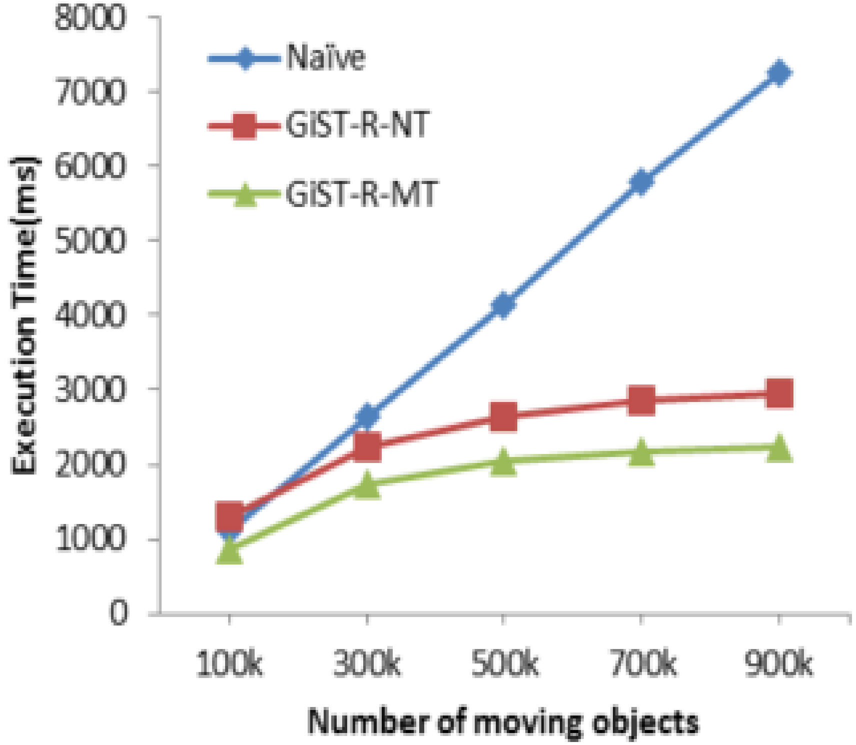
- **Trajectory Data : T-Drive**
  - Microsoft Research Asia, Beijin

- **6-month real dataset of 30,000 taxis in Beijing**
  - Total distance: almost 0.5 billion (446 million) KM
  - Number of GPS points: almost 1 billion (855 million)



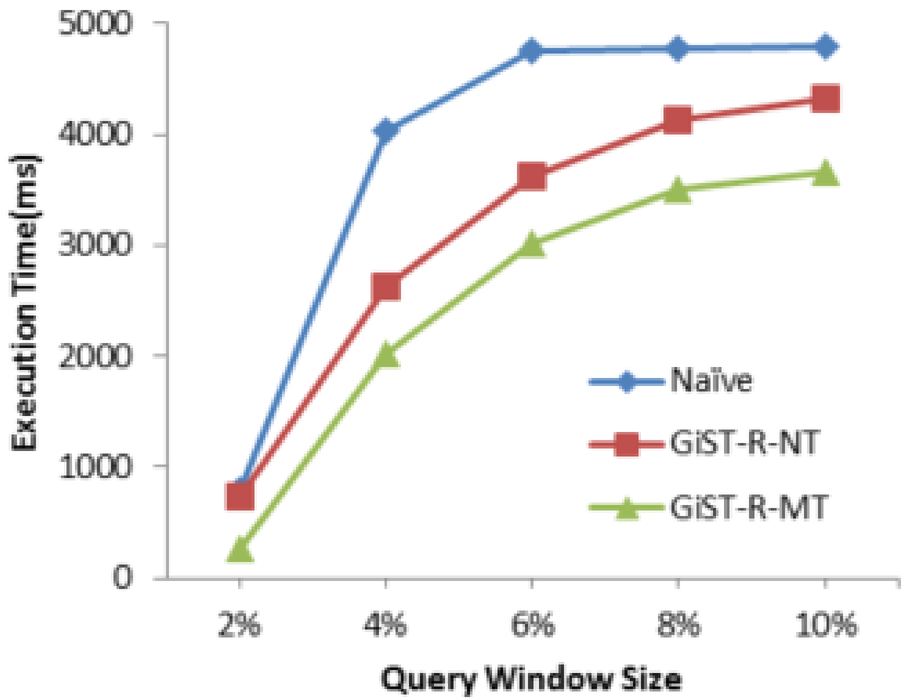
Performance Evaluation

- **Increasing Number of Moving Objects**



Performance Evaluation

- **Increasing Query Window Size**
  - for 500k moving objects



Conlusion

- **PostTrajectory**
  - https://github.com/awarematics/posttrajectory
- **Plan**
  - C++ Porting
  - Enhanced Spatiotemporal Join