pg routing을 활용한 주행거리 면적계산

작성자 : 정민기

1.pg_routing이란?

postgis의 노드-링크 데이터를 기반으로
지리공간 알고리즘 함수를 제공하는 extension

제공하는 알고리즘
- All Pairs Shortest Path, Johnson's Algorithm
- All Pairs Shortest Path, Floyd-Warshall Algorithm
- Shortest Path A*
- Bi-directional Dijkstra Shortest Path
- Bi-directional A* Shortest Path
- Shortest Path Dijkstra
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Traveling Salespreson Problem
- Turn Restriction Shortest Path (TRSP)

2.노드-링크 데이터란?

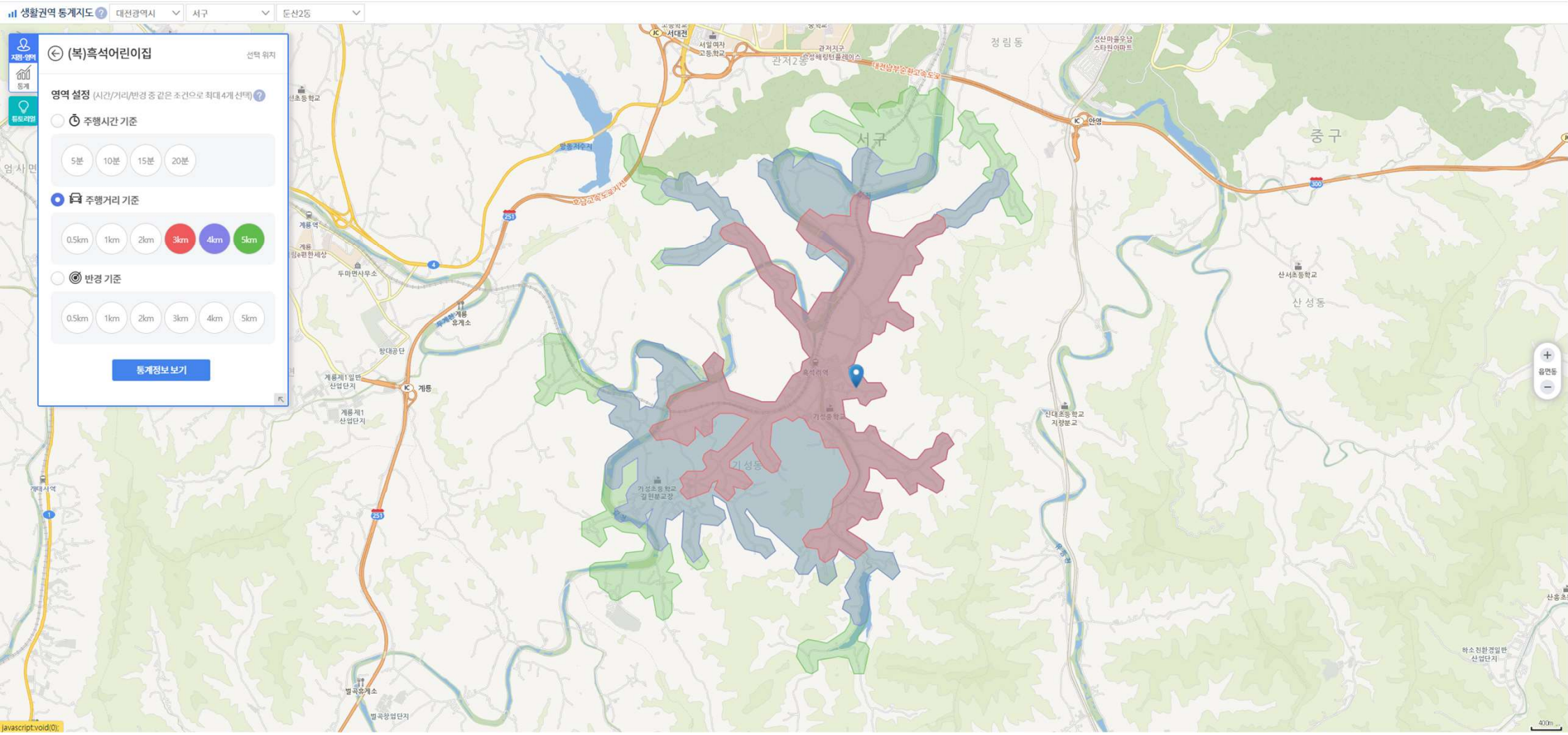
1) 노드 – 교통관점에서 차량이 도로를 주행함에 있어 속도의 변화가 발생되는 지점을 표현 ex) 교차로, 고가도로의 시종점

2) 링크 – 교통관점에서 링크는 노드와 노드를 연결한 선을 의미하고 실제 도로를 표현

국토 교통부에서 제공하는 국가 교통정보 표준 노드-링크 예시

3.Driving-Distance 사용예시

SGIS(통계청)의 생활권역 통계지도



4.Driving-Distance 함수

Parameters

Column	Type	Description
edges_sql	`TEXT`	SQL query as described above.
start_vid	`BIGINT`	Identifier of the starting vertex.
start_vids	`ARRAY[ANY-INTEGER]`	Array of identifiers of the starting vertices.
distance	`FLOAT`	Upper limit for the inclusion of the node in the result.
directed	`BOOLEAN`	(optional). When `false` the graph is considered as Undirected. Default is `true` which considers the graph as Directed.
equicost	`BOOLEAN`	(optional). When `true` the node will only appear in the closest `start_vid` list. Default is `false` which resembles several calls using the single starting point signatures. Tie brakes are arbitrary.

Inner query

Column	Type	De-fault	Description
id	`ANY-INTEGER`		Identifier of the edge.
source	`ANY-INTEGER`		Identifier of the first end point vertex of the edge.
target	`ANY-INTEGER`		Identifier of the second end point vertex of the edge.
cost	`ANY-NUMERICAL`		Weight of the edge *(source, target)* • When negative: edge *(source, target)* does not exist, therefore it's not part of the graph.
reverse_cost	`ANY-NUMERICAL`	-1	Weight of the edge *(target, source)*, • When negative: edge *(target, source)* does not exist, therefore it's not part of the graph.

```
SELECT * FROM pgr_drivingDistance(
        'SELECT id, source, target, cost, reverse_cost FROM edge_table',
        2, 3
      );
 seq | node | edge | cost | agg_cost
-----+------+------+------+----------
   1 |    2 |   -1 |    0 |        0
   2 |    1 |    1 |    1 |        1
   3 |    5 |    4 |    1 |        1
   4 |    6 |    8 |    1 |        2
   5 |    8 |    7 |    1 |        2
   6 |   10 |   10 |    1 |        2
   7 |    7 |    6 |    1 |        3
   8 |    9 |    9 |    1 |        3
   9 |   11 |   12 |    1 |        3
  10 |   13 |   14 |    1 |        3
(10 rows)
```

Result Columns

Returns set of `(seq [, start_v], node, edge, cost, agg_cost)`

Column	Type	Description
seq	`INTEGER`	Sequential value starting from **1**.
start_vid	`INTEGER`	Identifier of the starting vertex.
node	`BIGINT`	Identifier of the node in the path within the limits from `start_vid`.
edge	`BIGINT`	Identifier of the edge used to arrive to `node`. `0` when the `node` is the `start_vid`.
cost	`FLOAT`	Cost to traverse `edge`.
agg_cost	`FLOAT`	Aggregate cost from `start_vid` to `node`.

5.Driving-Distance Inner Query(edges_sql)

```
SELECT
    ST_MAKEPOLYGON(
        ST_EXTERIORRING(
            ST_UNION(
                ST_BUFFER(ml.geom, 150, 2)
            )
        )
    )
FROM
(
    SELECT
        *
    FROM
        PGR_DRIVINGDISTANCE(
            'SELECT
                link_id::bigint as id,
                f_node::bigint as source,
                t_node::bigint as target,
                length::double precision AS cost
             FROM
                hm_new.moct_link ''::text
            , 1210023900
            , 3000::double precision
            , false
        )
) dd
, hm_new.moct_link ml
WHERE
    ml.t_node = dd.node::character varying
```

Inner query :: edges_sql(해당 함수가 링크테이블을 인식할 수 있게 형변환 및 별칭을 부여)

id : 링크의 id

source : 시작노드의 id

target : 도착노드의 id

cost : 링크의 거리

링크 데이터가 있는 테이블

Inner query

Column	Type	De-fault	Description
id	ANY-INTEGER		Identifier of the edge.
source	ANY-INTEGER		Identifier of the first end point vertex of the edge.
target	ANY-INTEGER		Identifier of the second end point vertex of the edge.
cost	ANY-NUMERICAL		Weight of the edge *(source, target)* • When negative: edge *(source, target)* does not exist, therefore it's not part of the graph.
reverse_cost	ANY-NUMERICAL	-1	Weight of the edge *(target, source)*, • When negative: edge *(target, source)* does not exist, therefore it's not part of the graph.

6.Driving-Distance 함수의 사용

```
SELECT
    ST_MAKEPOLYGON(
        ST_EXTERIORRING(
            ST_UNION(
                ST_BUFFER(ml.geom, 150, 2)
            )
        )
    )
FROM
(
    SELECT
        *
    FROM
        PGR_DRIVINGDISTANCE(
            ' SELECT
                  link_id::bigint as id,
                  f_node::bigint as source,
                  t_node::bigint as target,
                  length::double precision AS cost
              FROM
                  hm_new.moct_link '::text
            , 1210023900
            , 3000::double precision
            , false
        )
) dd
, hm_new.moct_link ml
WHERE
    ml.t_node = dd.node::character varying
```

edges_sql

start_vid (시작노드)

distance (주행거리)

결과

Parameters

Column	Type	Description
edges_sql	`TEXT`	SQL query as described above.
start_vid	`BIGINT`	Identifier of the starting vertex.
start_vids	`ARRAY[ANY-INTEGER]`	Array of identifiers of the starting vertices.
distance	`FLOAT`	Upper limit for the inclusion of the node in the result.
directed	`BOOLEAN`	(optional). When `false` the graph is considered as Undirected. Default is `true` which considers the graph as Directed.
equicost	`BOOLEAN`	(optional). When `true` the node will only appear in the closest `start_vid` list. Default is `false` which resembles several calls using the single starting point signatures. Tie brakes are arbitrary.

	seq integer	node bigint	edge bigint	cost double precision	agg_cost double precision
1	1	1210023900	-1	0	0
2	2	1210025300	1210013304	111.8513	111.8513
3	3	1210027100	1210017702	103.7615	215.6128
4	4	1210026300	1210016302	115.6212	227.4725
5	5	1190003200	1190008900	250.9028	250.9028
6	6	1210020000	1210013203	336.4448	336.4448
7	7	1210027300	1210048000	110.7504	338.2229
8	8	1210027101	1210017705	133.3591	348.9719
9	9	1210020100	1210042100	49.7446	386.1894
10	10	1190003000	1190008700	166.9347	417.8375
11	11	1210026400	1210016303	230.5065	457.979
12	12	1210020101	1210044801	150.6047	536.7941
13	13	1190026400	1190050200	196.9049	545.8768
14	14	1210028700	1210017706	217.8172	566.7891
15	15	1210028300	1210049200	283.6002	621.8231
16	16	1210028500	1210049400	82.762	704.5851
17	17	1190002900	1190008300	292.1537	709.9912
18	18	1210029500	1210017704	156.4551	723.2442
19	19	1210021201	1210045101	272.2436	730.2226
20	20	1210027200	1210016304	329.0968	787.0758
21	21	1210018500	1210013202	458.396	794.8408
22	22	1210080200	1210162600	263.7285	800.5226
23	23	1210028800	1210050000	121.1102	825.6953
24	24	1210018300	1210041600	477.2369	863.4263
25	25	1210021200	1210045102	138.1973	868.4199
26	26	1190003700	1190010000	452.7105	870.548
27	27	1200013300	1210013201	77.8956	872.7364
28	28	1210018100	1210041000	37.8716	901.2979

7.Driving-Distance결과로 면적 구하기

```
SELECT
    ST_MAKEPOLYGON(
        ST_ExteriorRing(
            ST_UNION(
                ST_BUFFER(ml.geom, 150, 2)
            )
        )
    )
FROM
(
    SELECT
        *
    FROM
        pgr_drivingdistance( ~~ )
) dd
, hm_new.moct_link ml
WHERE
    ml.t_node = dd.node::character varying
```

drivingdistance 결과의 노드와
링크테이블의 도착지점 노드를 조인하여
geometry 정보를 획득