Spatial data processing

CS594: Big Data Visualization & Analytics

Fabio Miranda

https://fmiranda.me

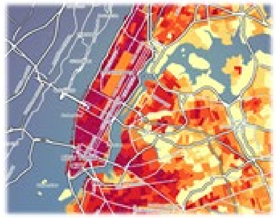
COMPUTER SCIENCE

Overview

- Spatial data
- Spatial queries
- Spatial indices

Spatial data

flickr



twitter

Infrastructure | Environment | Social media

Spatial data

- Spatial attributes
  - 2D: (x, y)
  - 3D: (x, y, z)
- Spatial data primitives:
  - Points
  - Lines
  - Polygons
- Other attributes:
  - Time
  - …

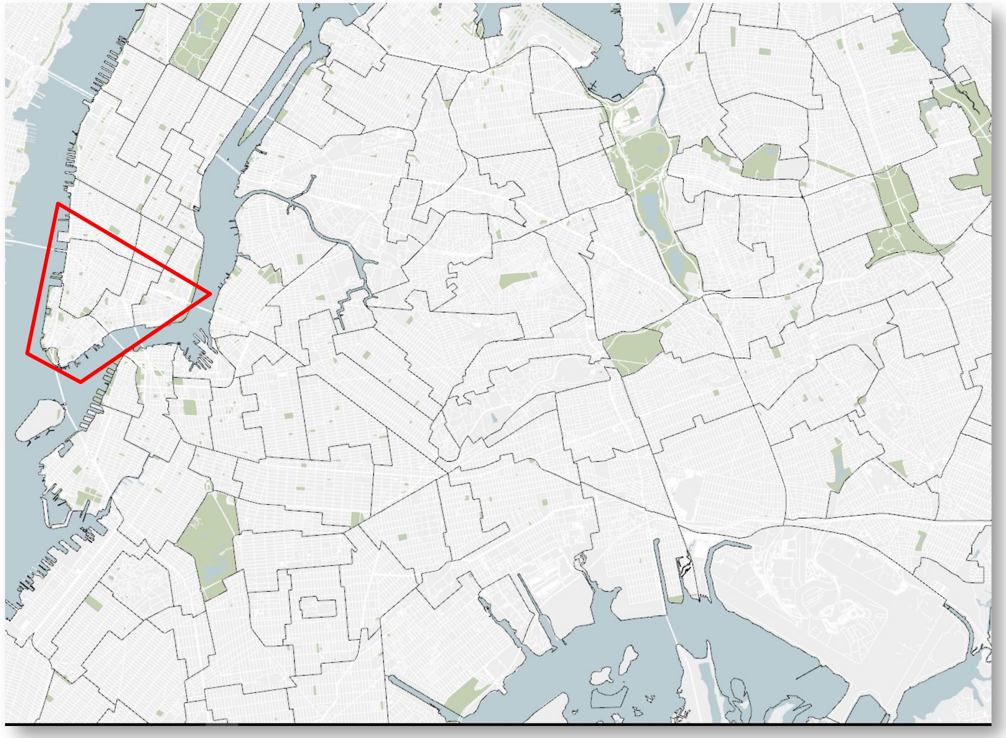
Spatial queries

- Spatial selection
- Spatial joins
- Spatial aggregation

Spatial selection queries

Select spatial objects that satisfy a spatial constraint.

```
SELECT *
FROM taxi T
WHERE T.pickup inside Lower Manhattan
```



Spatial selection queries

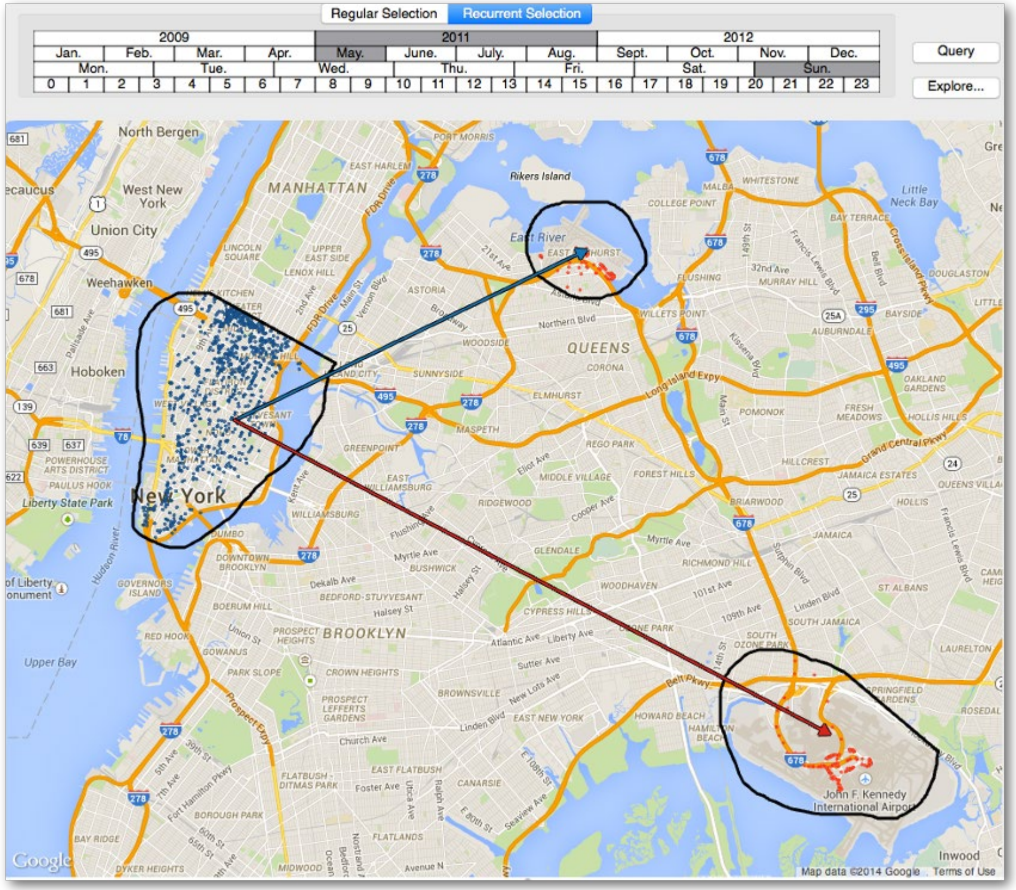
Select spatial objects that satisfy a spatial constraint.

```
SELECT *
FROM taxi T
WHERE T.pickup inside Lower Manhattan
AND (T.dropoff inside JFK OR
T.dropoff inside LGA)
```

Spatial selection queries

Select spatial objects that satisfy a spatial constraint and other constraints.

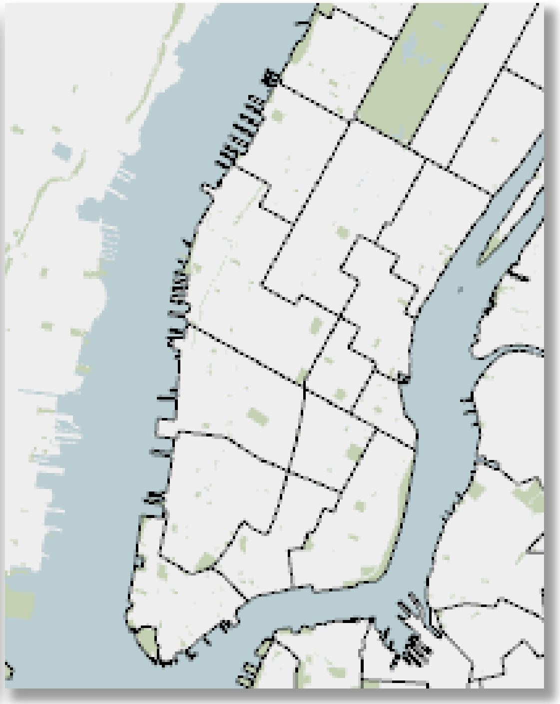
```
SELECT *
FROM taxi T
WHERE T.pickup inside Lower Manhattan
AND (T.dropoff inside JFK OR
T.dropoff inside LGA)
AND T.picktime in May 2011
```



Spatial joins

Select pairs of spatial objects satisfying a spatial constraint.

```
SELECT *
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
```



Spatial aggregation queries

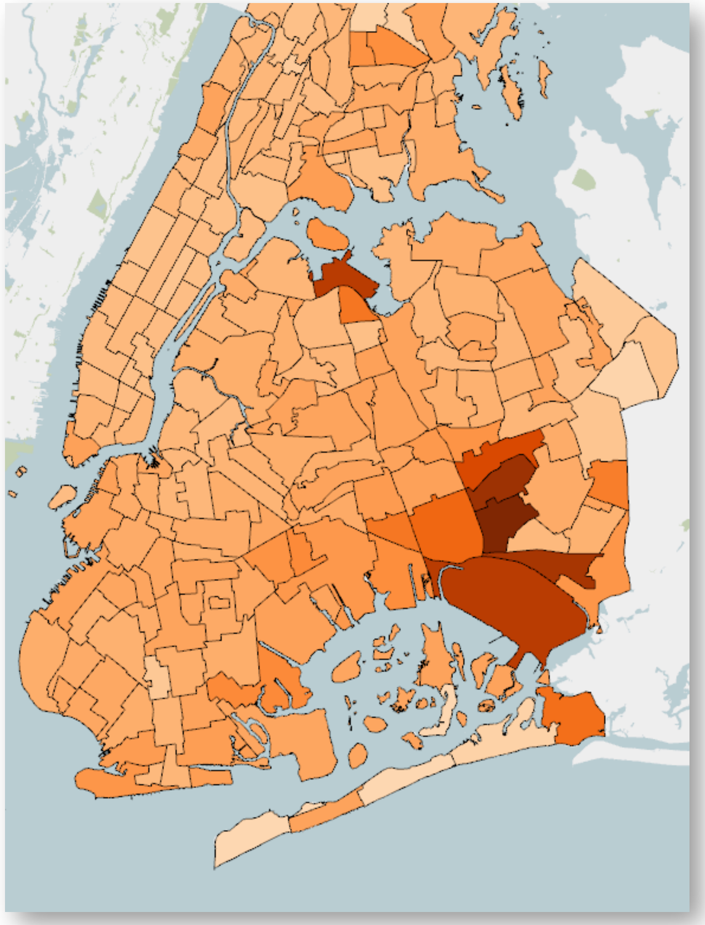
Aggregate data points over spatial regions.

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```

Spatial aggregation queries

Aggregate data points over spatial regions.

```
SELECT AVG(T.duration)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



Nearest neighbor queries

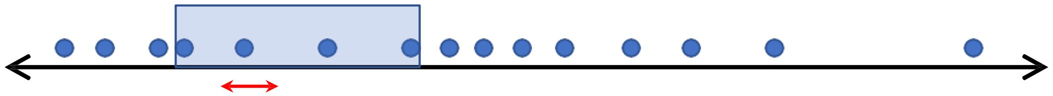
Find nearest points.

```
SELECT TOP(10)
FROM lots B, crime C
ORDER BY DISTANCE(B.geometry,
C.location)
```

Spatial index

- How to speedup these queries?
  - Spatial index!

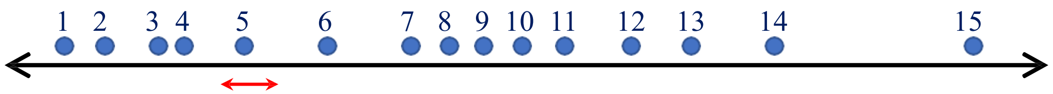
1-dimensional data



1-D range search: Find points between *x* and *y*

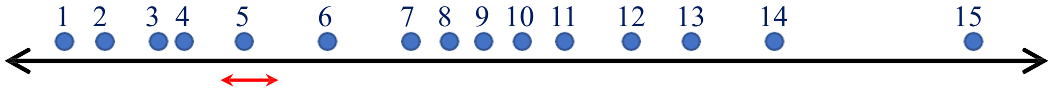
Unordered list: Fast insert O(1), slow range search O(n)
Ordered list: Slow insert O(n), binary search for x and y to do range search O(log n).

1-dimensional data



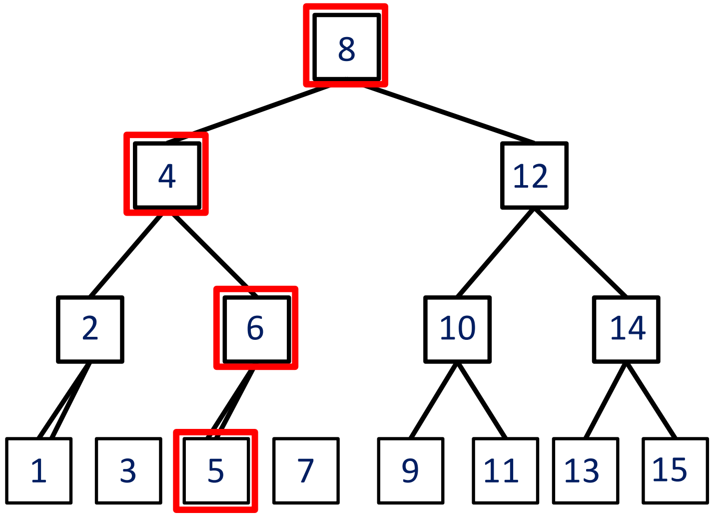
Find point with value 5

1-dimensional data: binary search

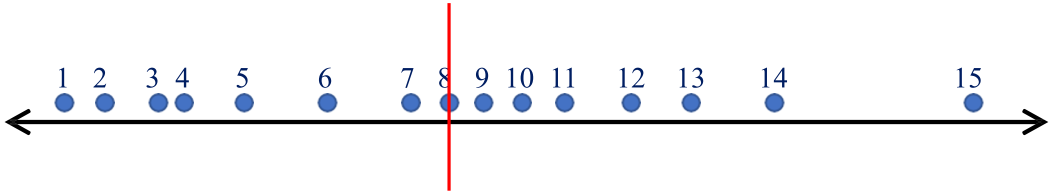


Find point with value 5

Binary Search Tree

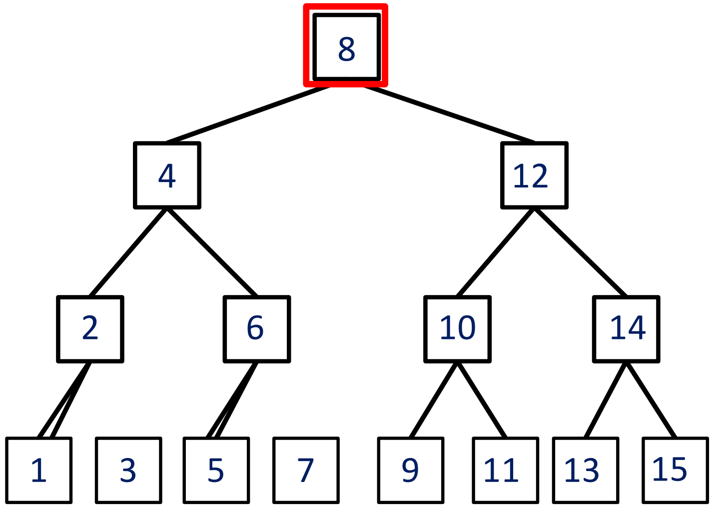


1-dimensional data: binary search

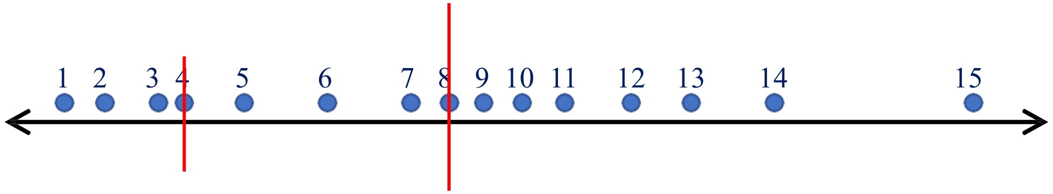


Find point with value 5

Binary Search Tree

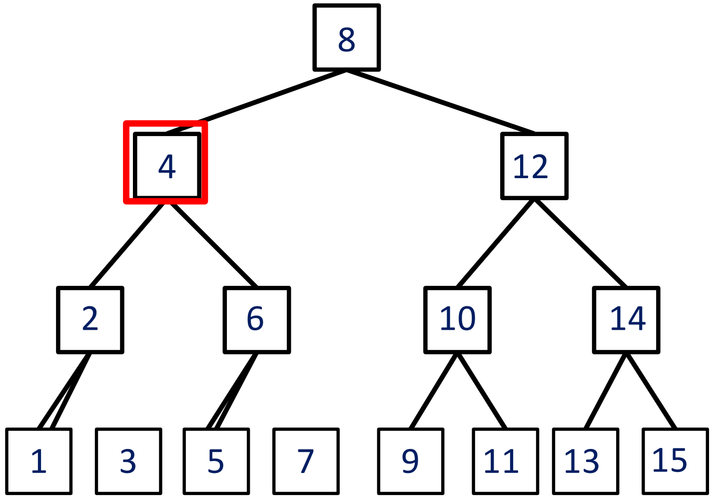


1-dimensional data: binary search

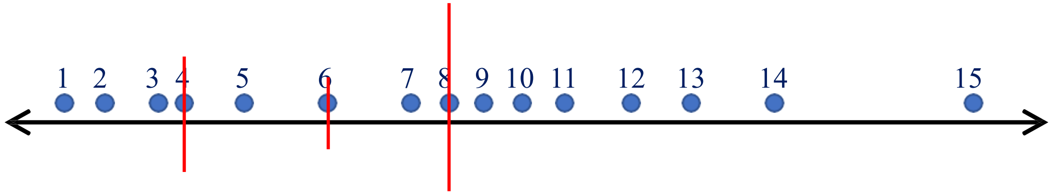


Find point with value 5

Binary Search Tree

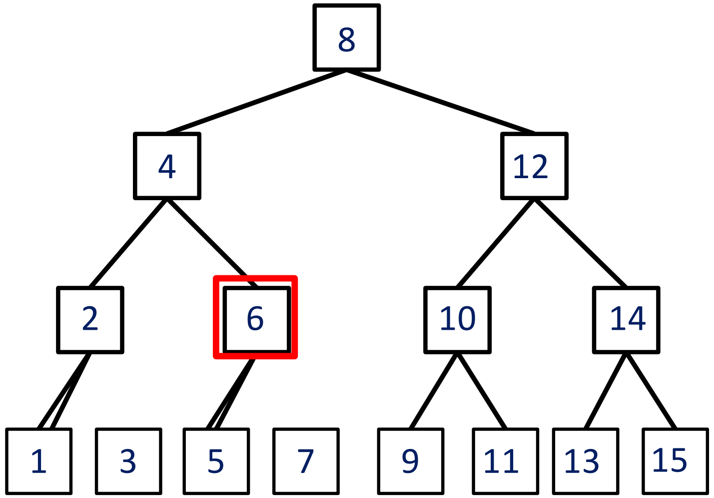


1-dimensional data: binary search

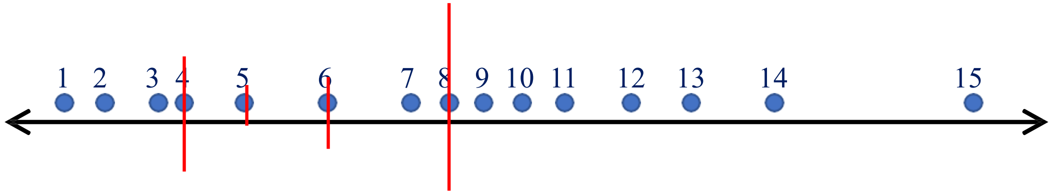


Find point with value 5

Binary Search Tree

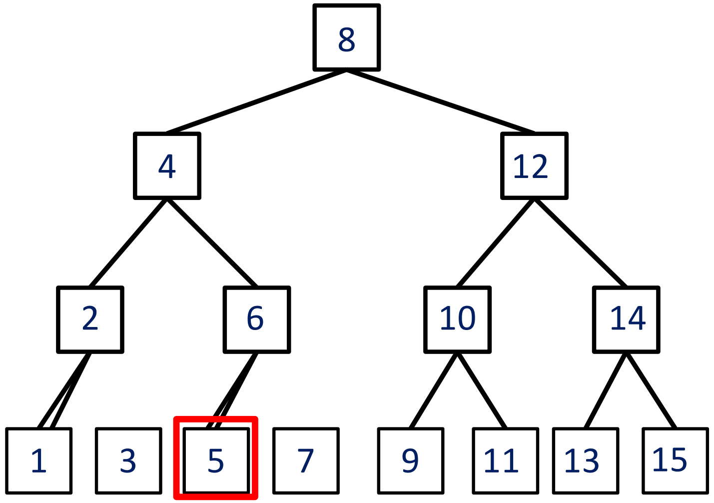


1-dimensional data: binary search

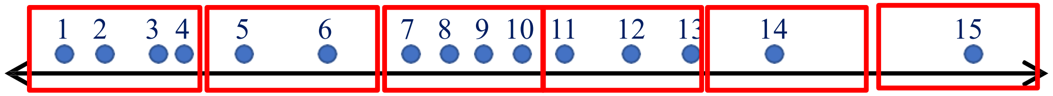


Find point with value 5

Binary Search Tree



1-dimensional data: hash function

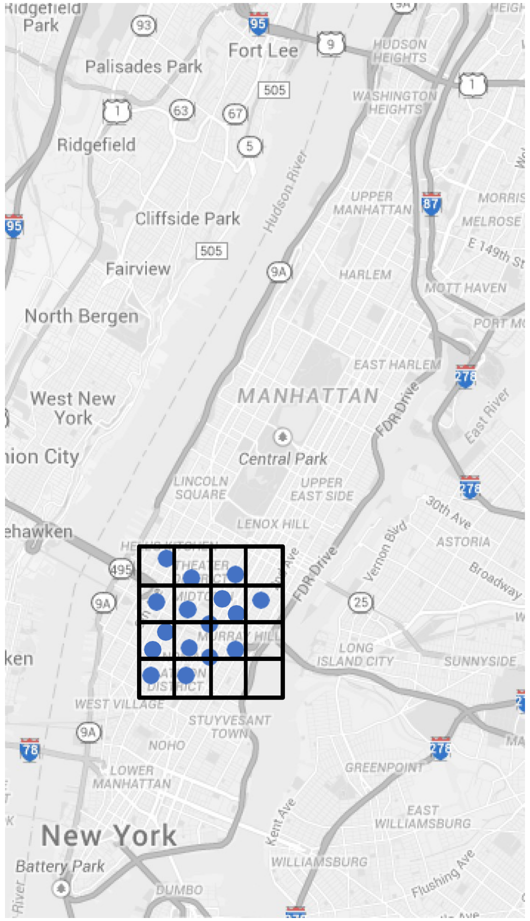


- Create bins using a hash function.
- Query:
  - Identify bin(s) satisfying query constraint.
  - Search within bin.

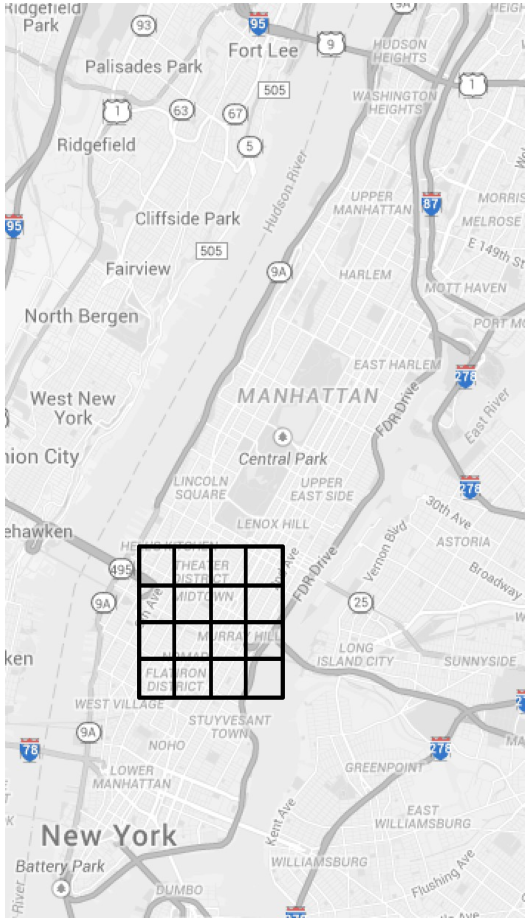
2-dimensional data

- Kd-Tree:
  - Extension of a binary search tree to higher dimensions.
  - Supports k-dimensional tree.
- Grid index:
  - Extension of hash index to higher dimensions.
  - Hash function is defined by a grid.
  - Overlay a grid covering the spatial region → assign objects to different grid cells.

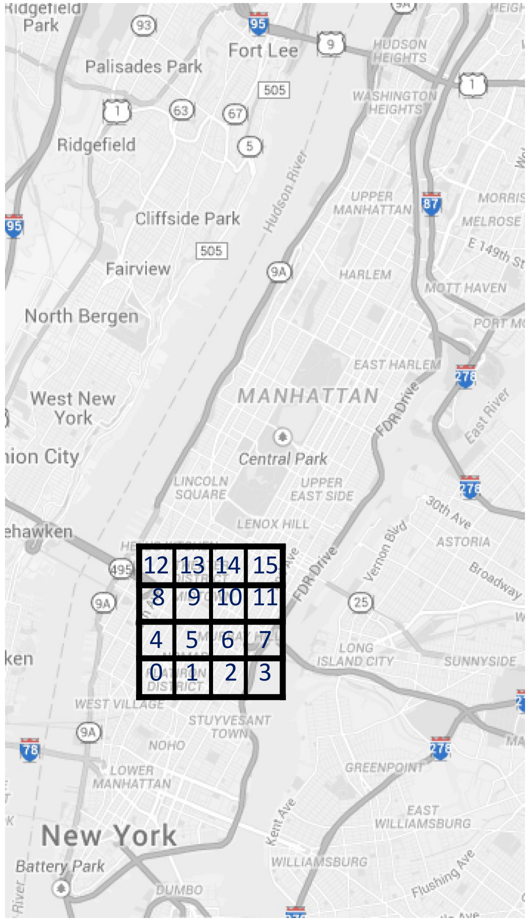
2-dimensional example



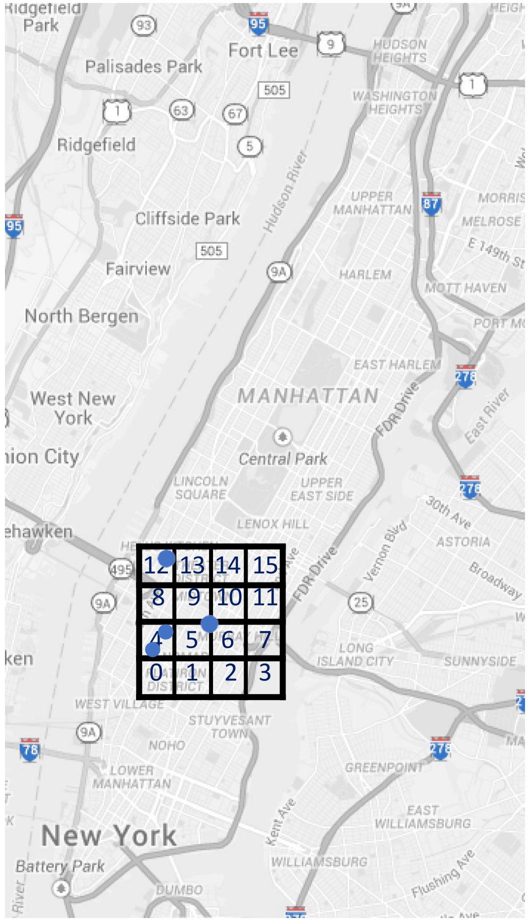
2-dimensional example



2-dimensional example

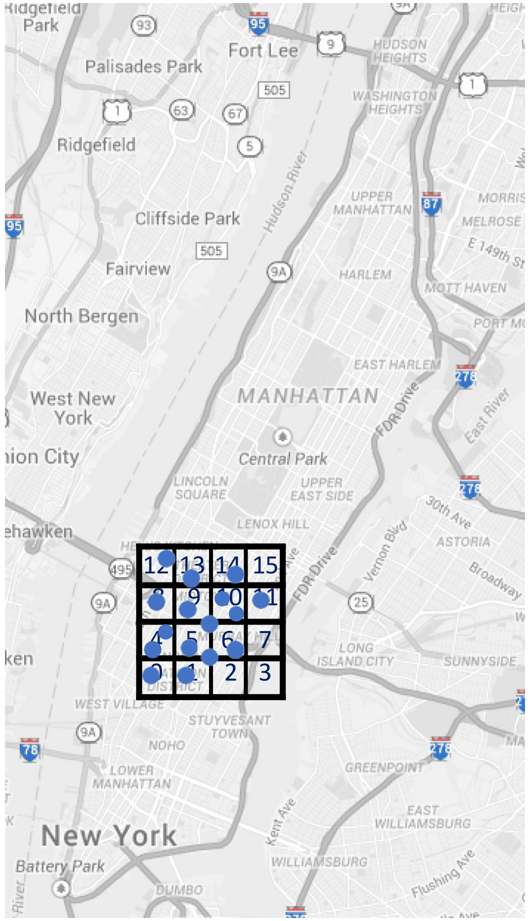


2-dimensional example



0	
1	
2	
3	
4	3 4
5	10
6	10
7	
8	
9	
10	
11	
12	1
13	
14	
15	

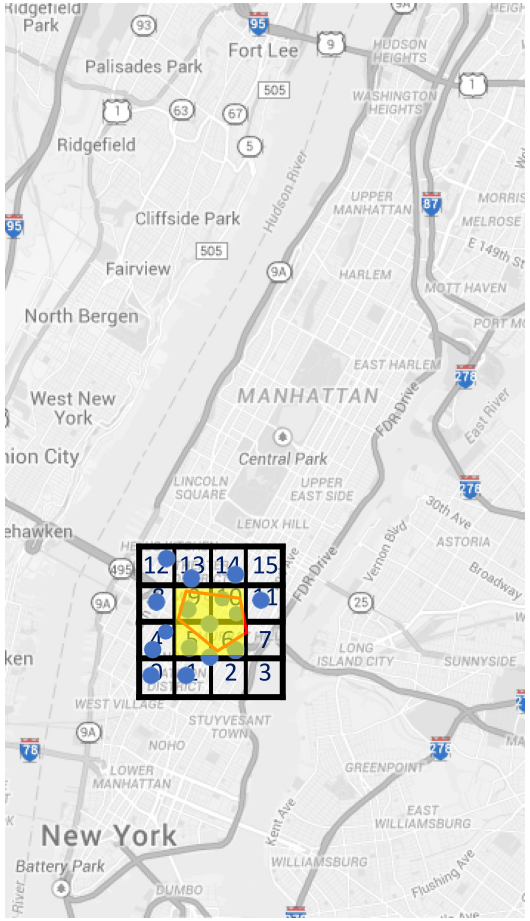
2-dimensional example



0	5
1	9 11
2	11
3	
4	3 4
5	8 10 11
6	10 11 15
7	
8	2
9	7
10	12 14
11	16
12	1
13	6
14	14
15	

2-dimensional example

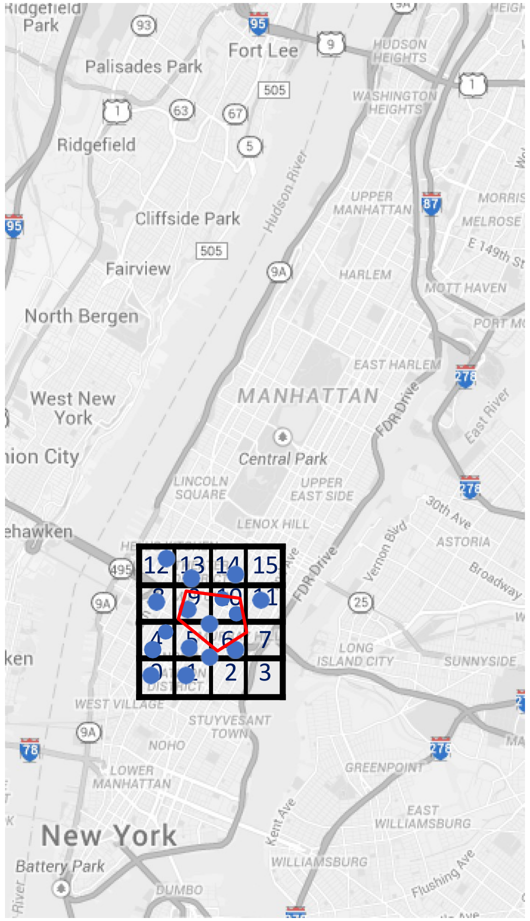
- Find Cells Intersected:
  - 5,6,9,10
- Test all points in these cells



0	5
1	9 11
2	11
3	
4	3 4
5	8 10 11
6	10 11 15
7	
8	2
9	7
10	12 14
11	16
12	1
13	6
14	14
15	

2-dimensional example

- Find Cells Intersected:
  - 5,6,9,10
- Test all points in these cells



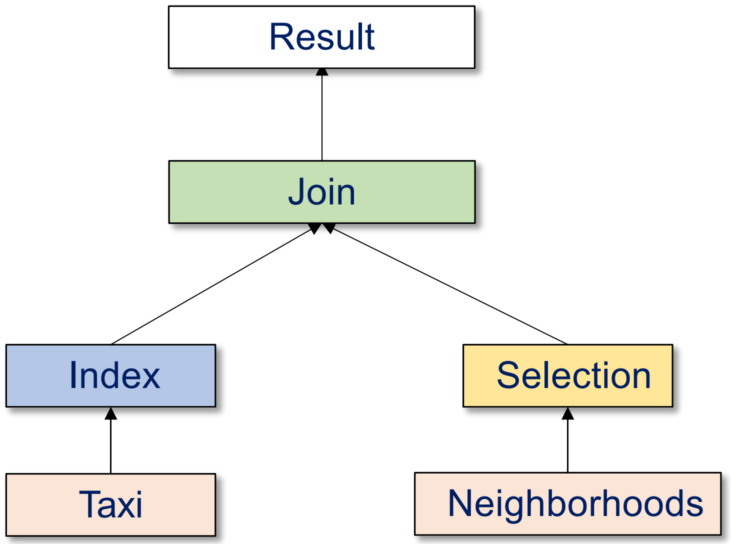
0	5
1	9 11
2	11
3	
4	3 4
5	8 10 11
6	10 11 15
7	
8	2
9	7
10	12 14
11	16
12	1
13	6
14	14
15	

Spatial queries

- Spatial selection
- Spatial joins
- Spatial aggregation

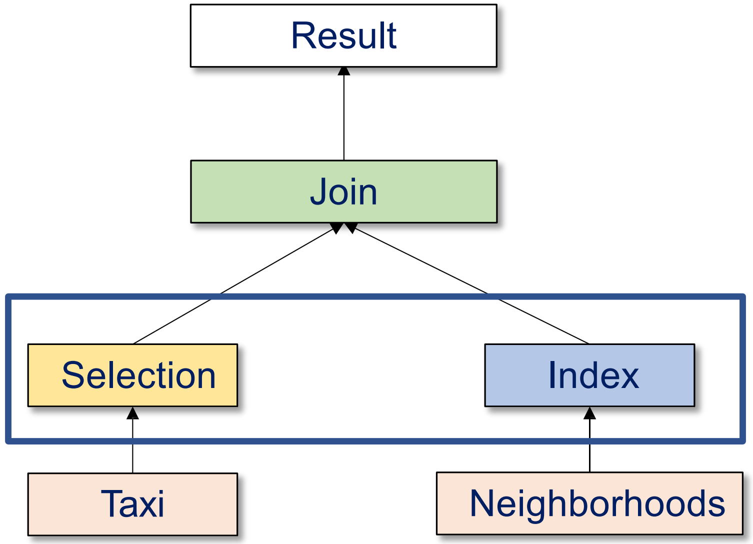
Spatial join: approach 1

- Create index
  - Trips
- For each neighborhood
  - Query for trips within that neighborhood using the index



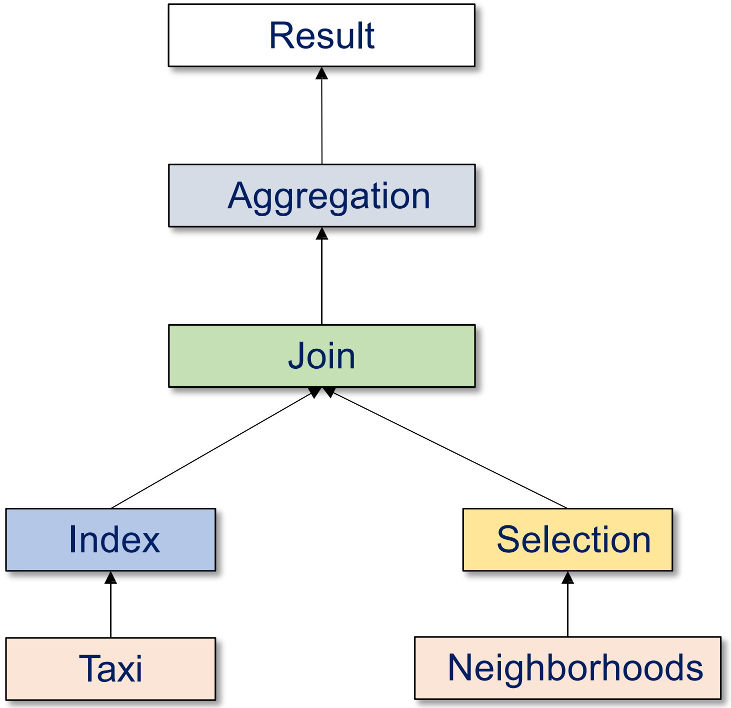
Spatial join: approach 2

- Create index
  - Neighborhoods
- For each trip
  - Query for neighborhood using the index
  - Add it to the corresponding neighborhood

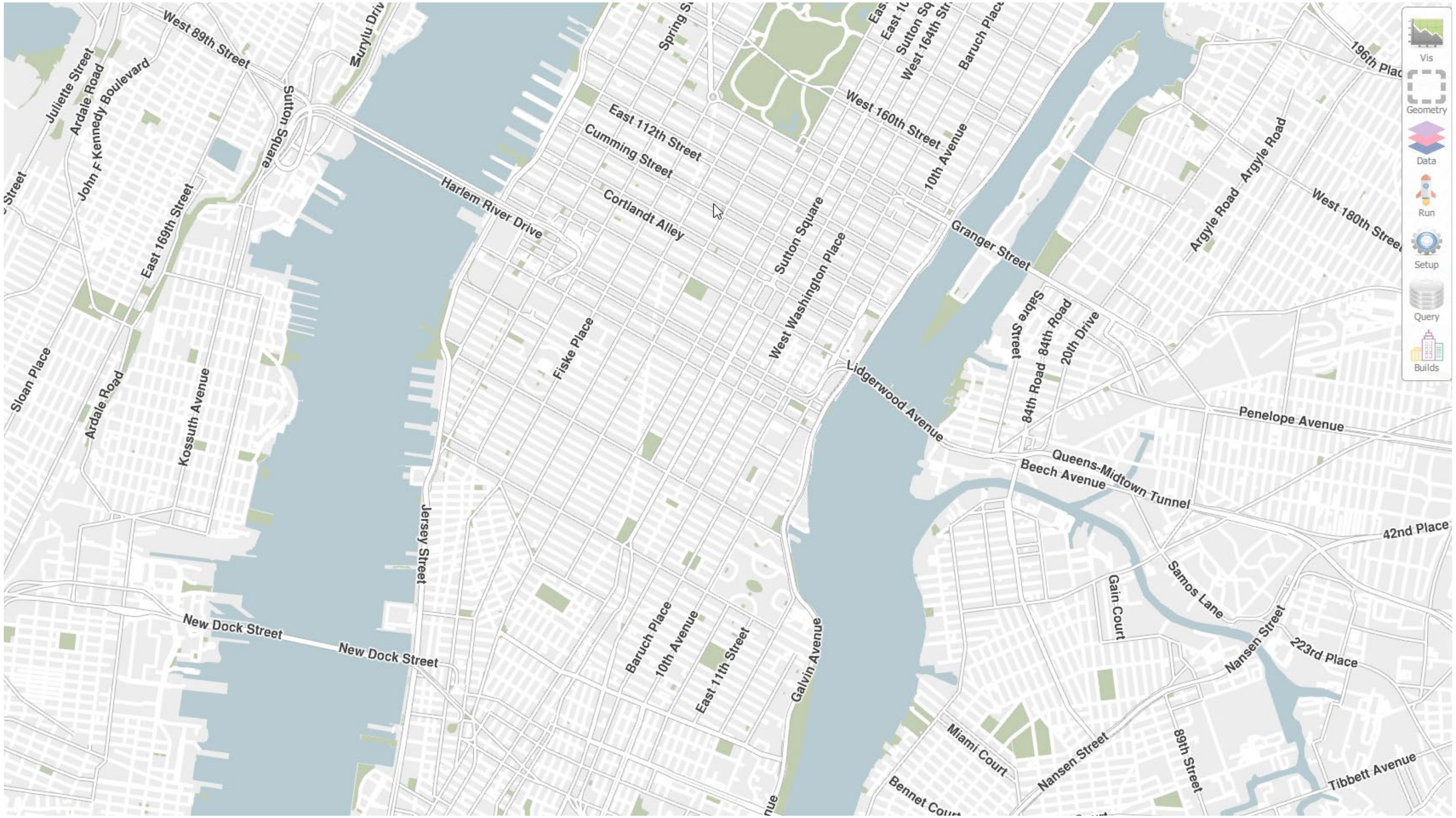


Spatial aggregation queries

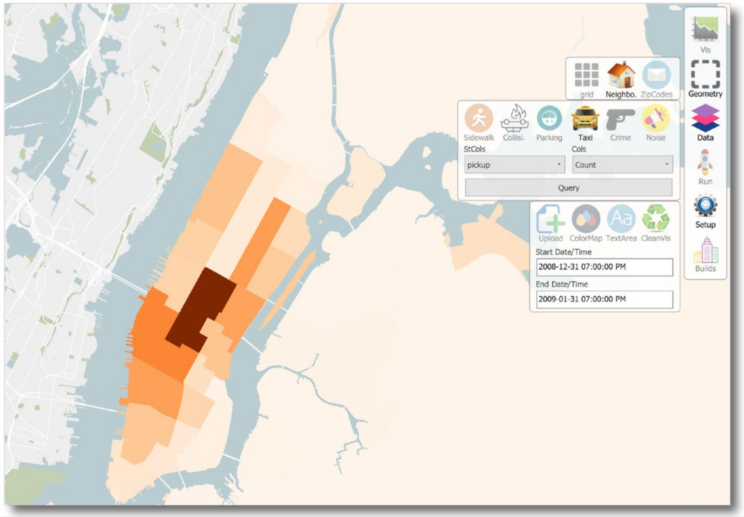
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



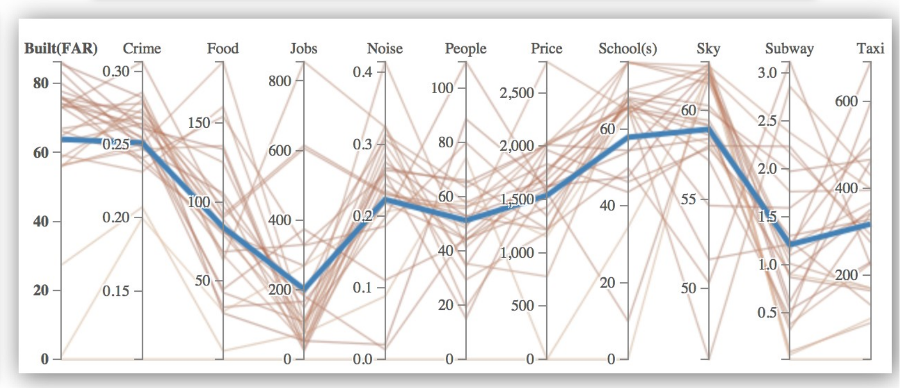
Visual queries



Usability through visual queries

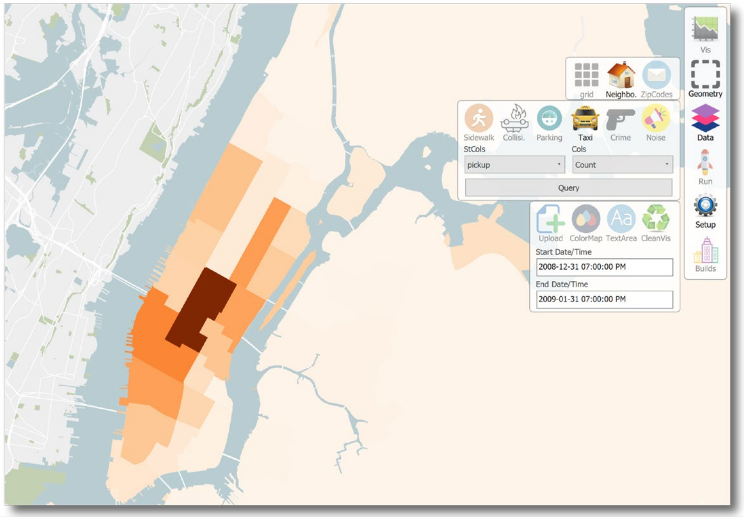


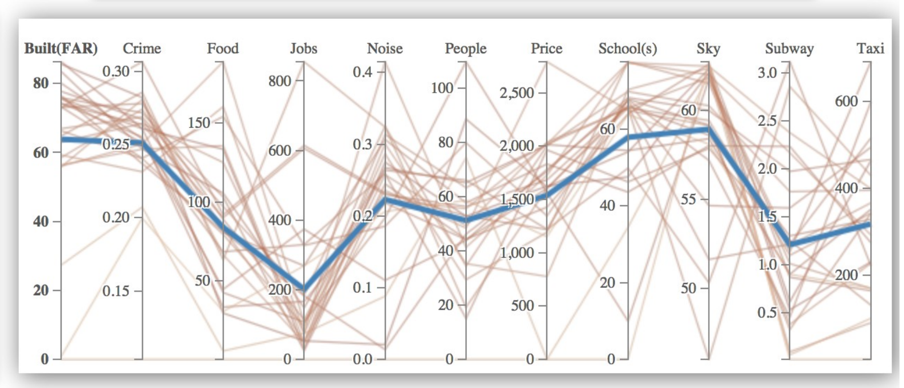

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
AND T.picktime in January 2016
GROUP BY N.id
```




```
SELECT COUNT(*)
FROM crime C, neighborhoods N
WHERE C.location INSIDE N.geometry
AND C.date in January 2016
GROUP BY N.id
```

Usability through visual queries






```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
AND T.picktime in January 2016
GROUP BY N.id
```

```
SELECT COUNT(*)
FROM crime C, neighborhoods N
WHERE C.location INSIDE N.geometry
AND C.date in January 2016
GROUP BY N.id
```

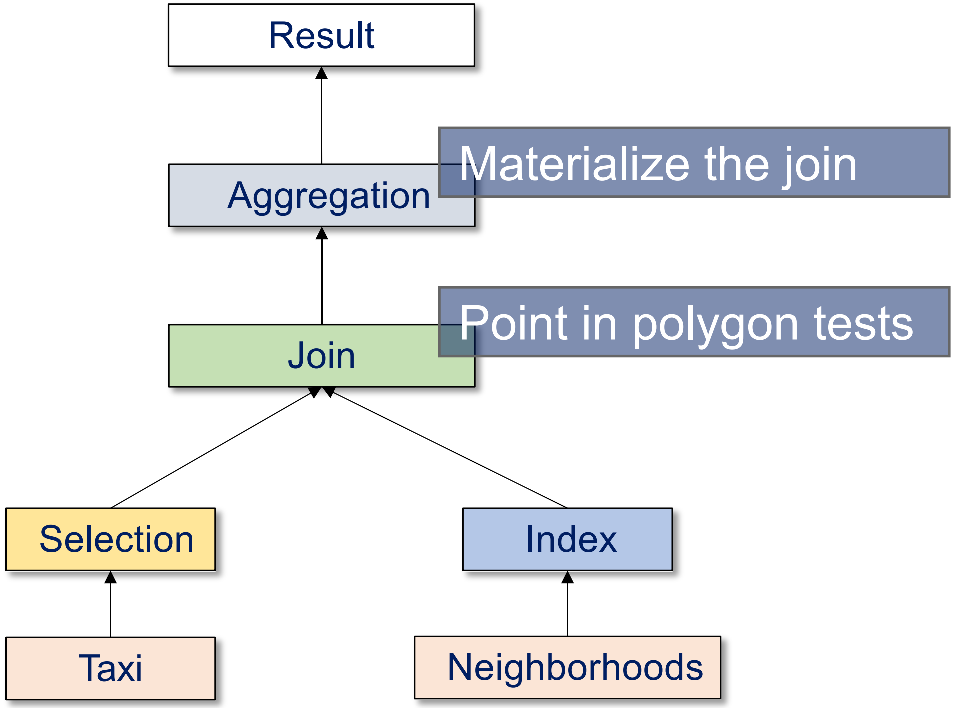
Crime
Food
Jobs
Noise
Price
Schools
Sky
exposure

Spatial aggregation queries

- Set of trips
  - Point data
  - ~340 million trips
- Set of neighborhoods
  - Polygon data
  - ~260 neighborhoods
- How to join these two data sets?

Spatial aggregation queries

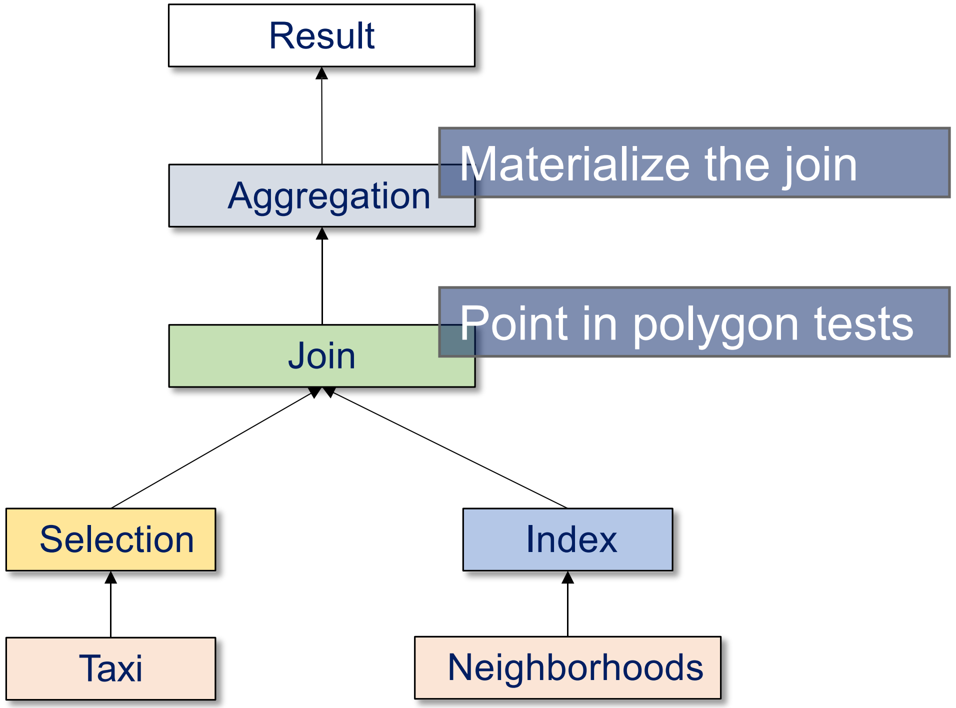
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



Spatial aggregation queries

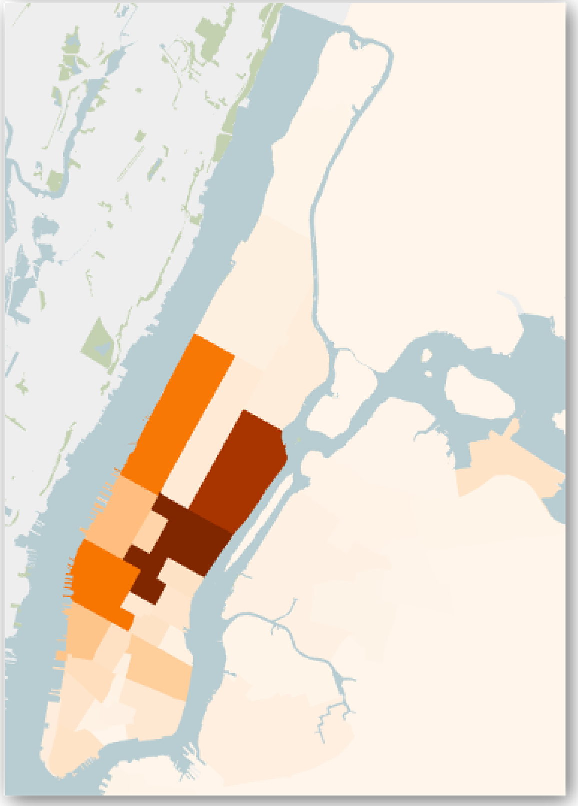
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```

- Existing spatial databases.
- Several minutes!



Spatial aggregation queries

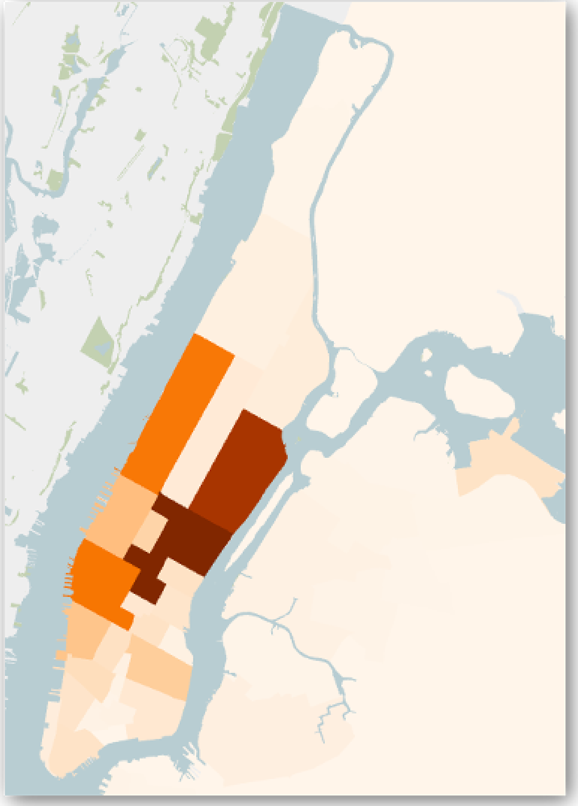
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



- Possible solution: pre-compute aggregation

Spatial aggregation queries

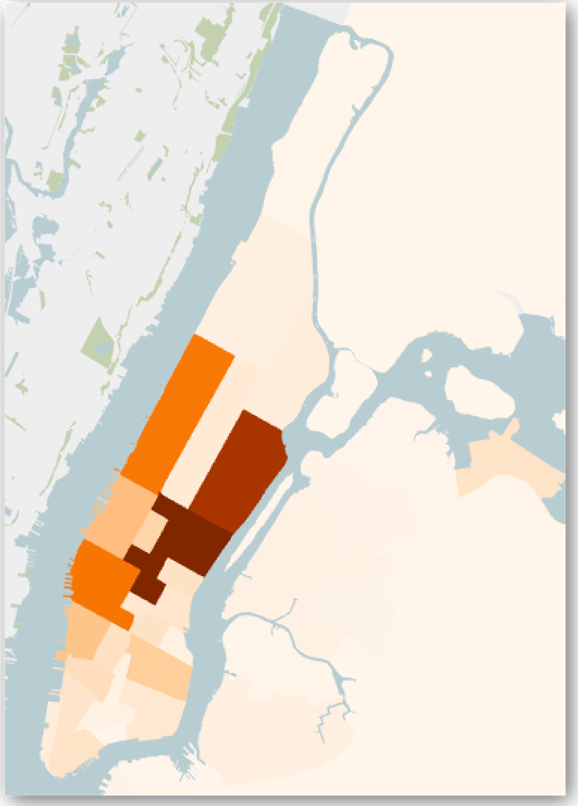
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
AND T.picktime in March 2011
GROUP BY N.id
```



- Possible solution: cube-based structures – nanocubes, …

Spatial aggregation queries

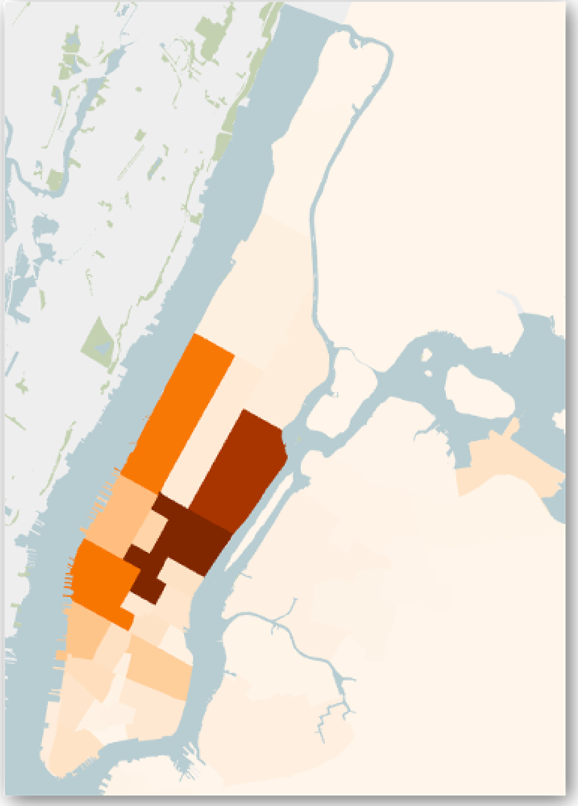
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
AND T.picktime in March 2011
AND T.duration > 10 minutes
GROUP BY N.id
```



- Possible solution: cube-based structures – nanocubes, …
- Space explosion!

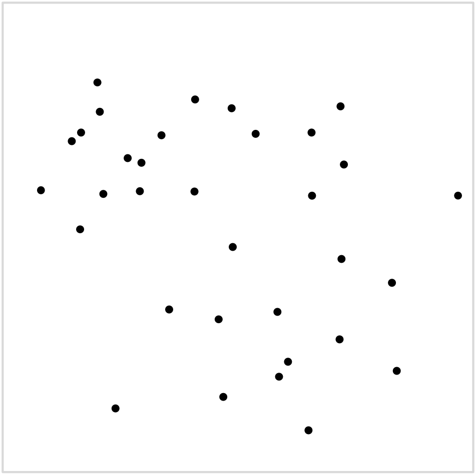
Desiderata

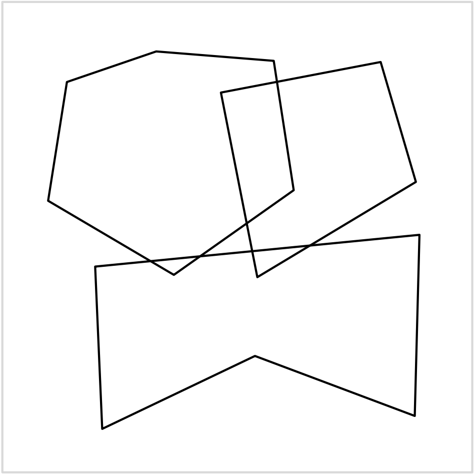
- Interactive response times
- Avoid costly preprocessing



Running example

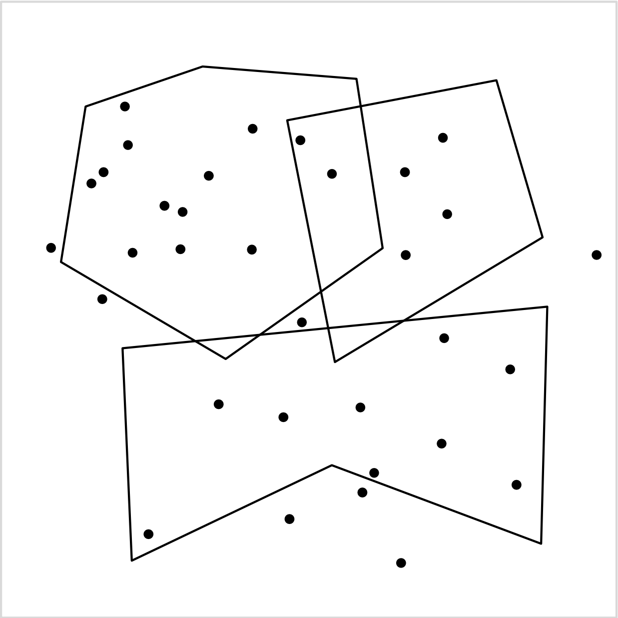
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```





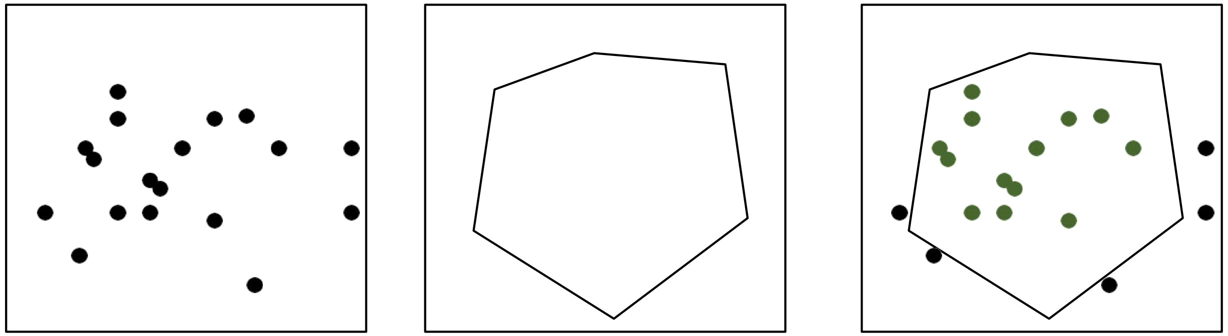
Running example

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



Spatial aggregation

- A geometric perspective of spatial aggregation.
- Spatial join: “drawing” points and polygons on the same canvas.



Input points   Input polygon   Spatial join

- Leverage the graphics pipeline of the GPU [Tzirita et al., 2017]

Raster Join

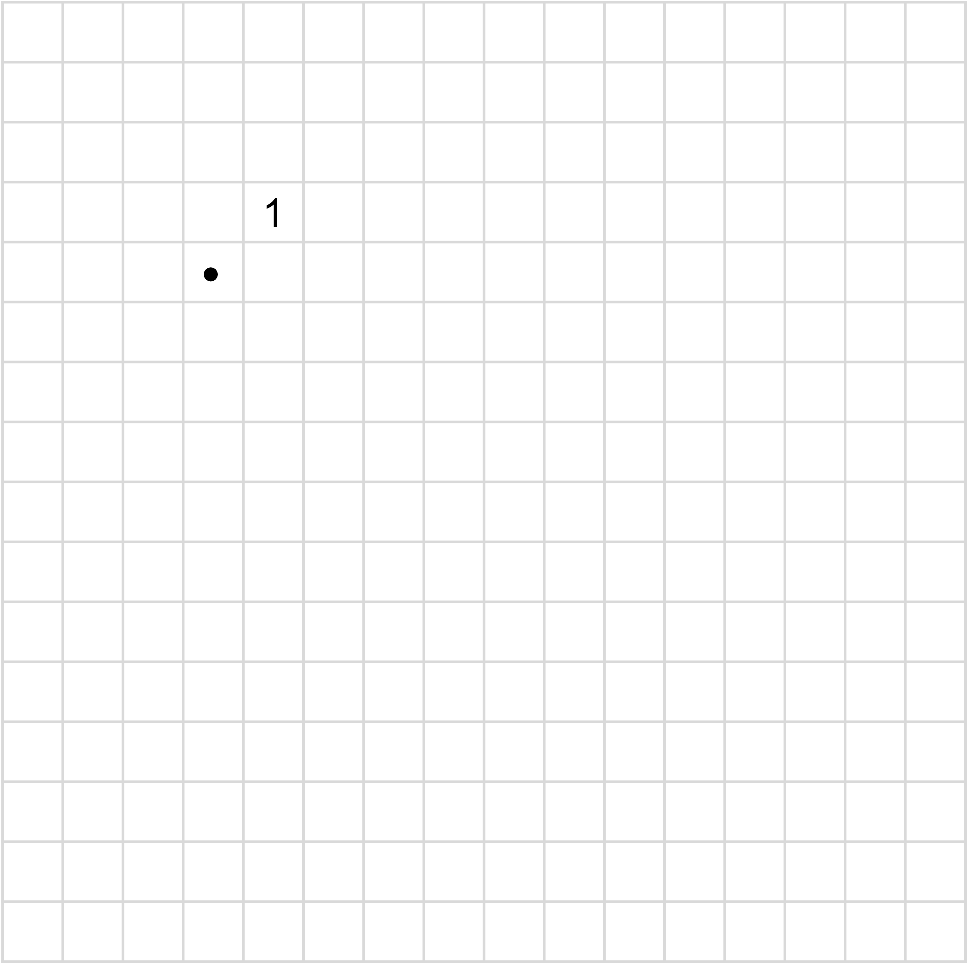
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```

Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```

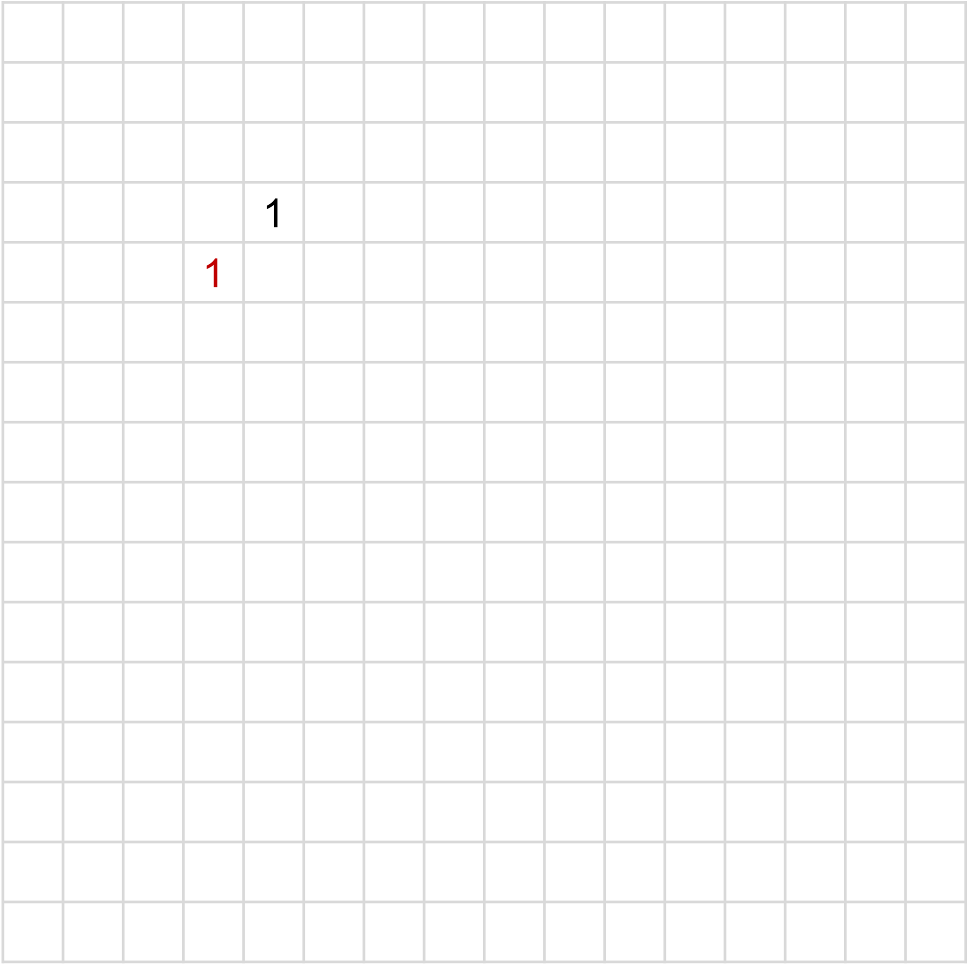
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



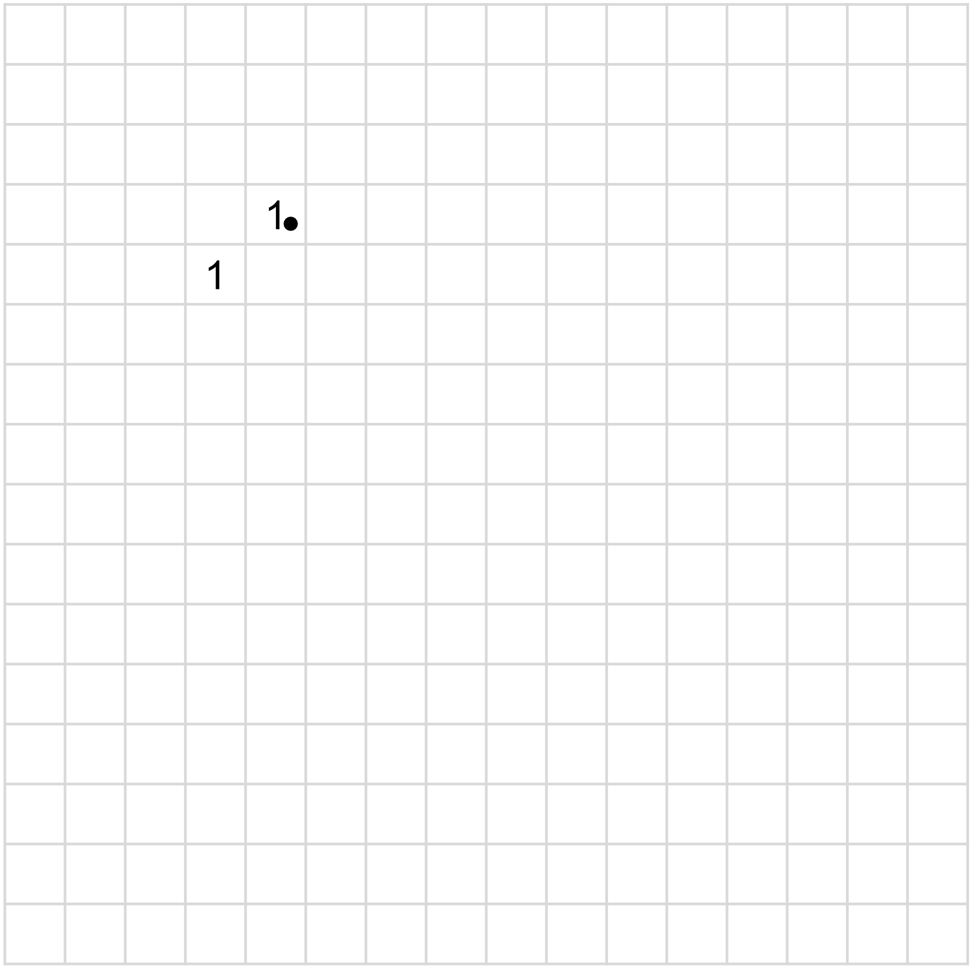
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



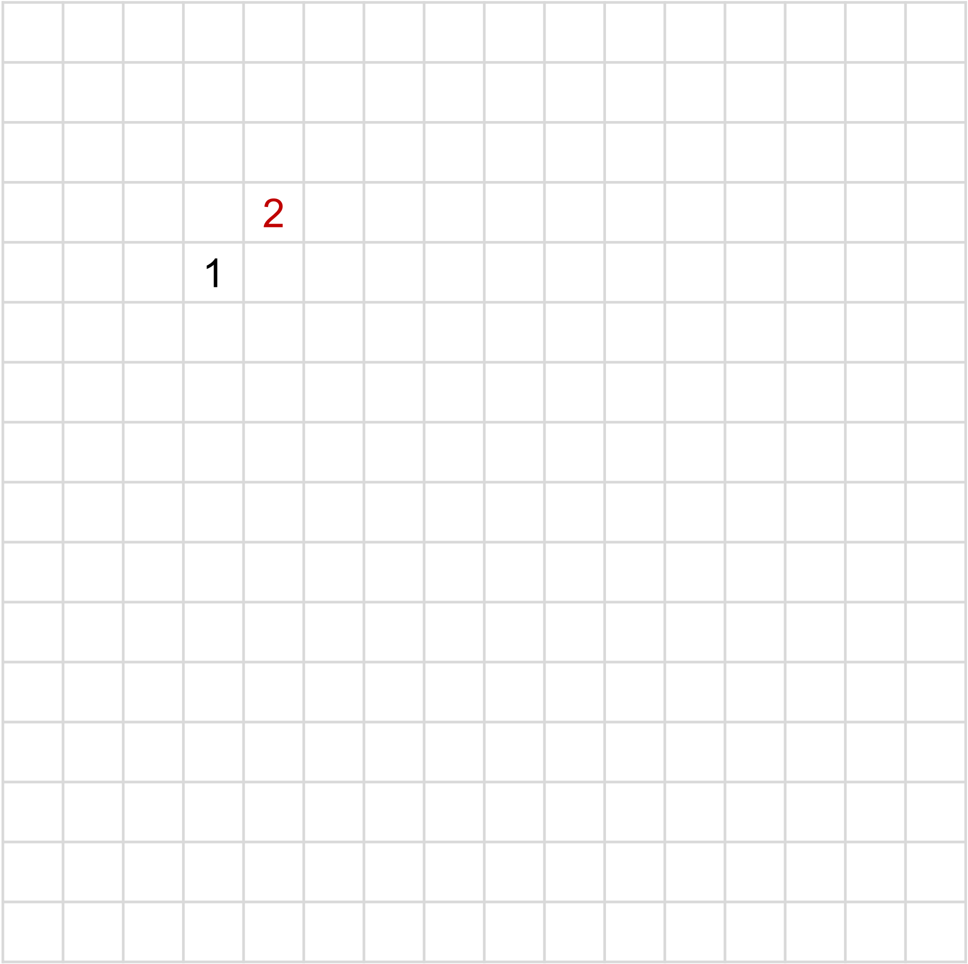
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



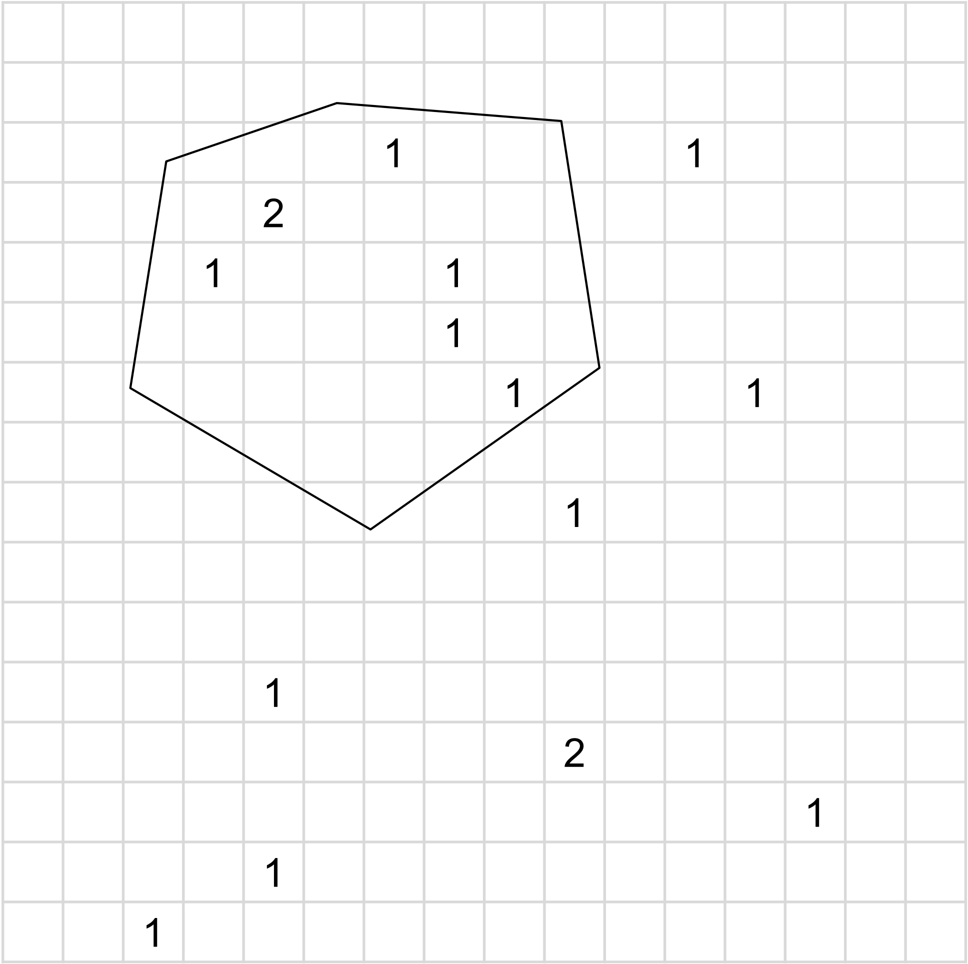
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



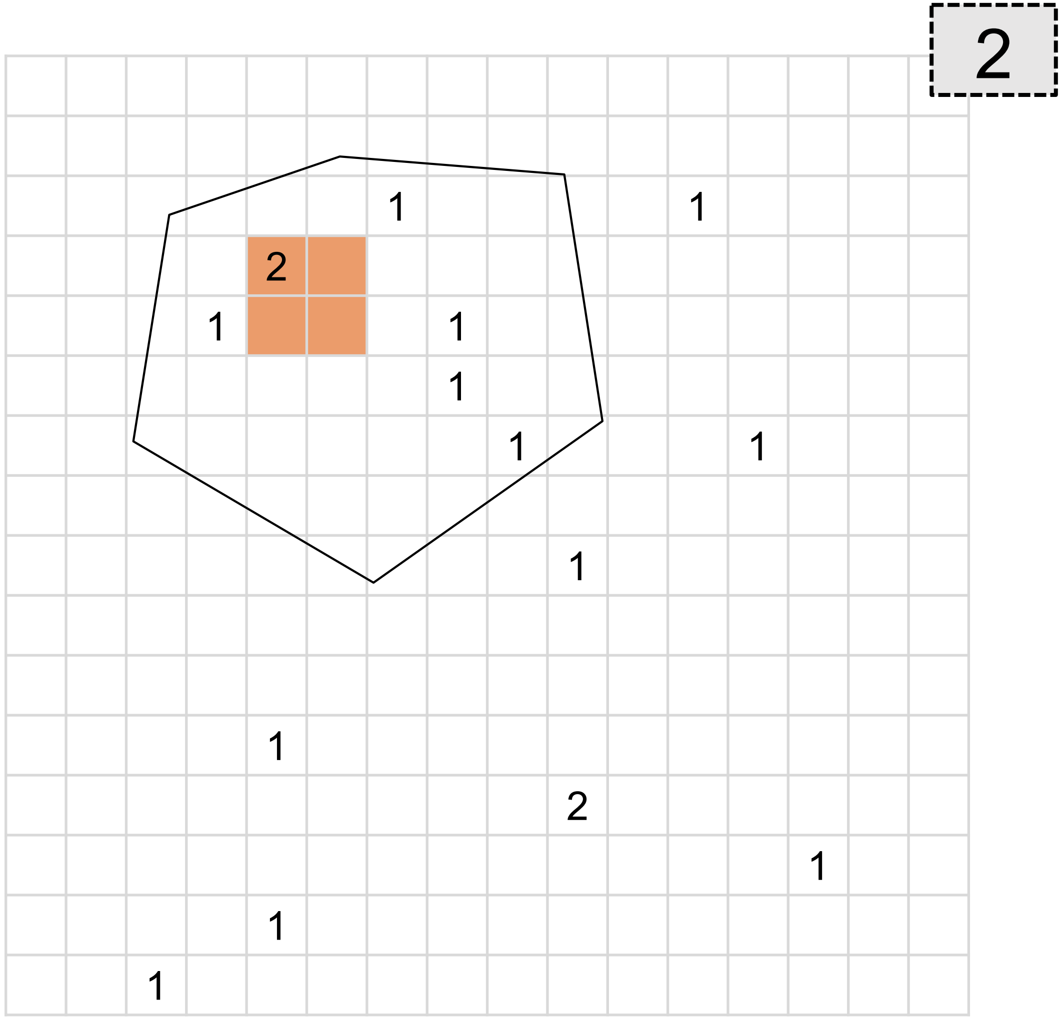
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



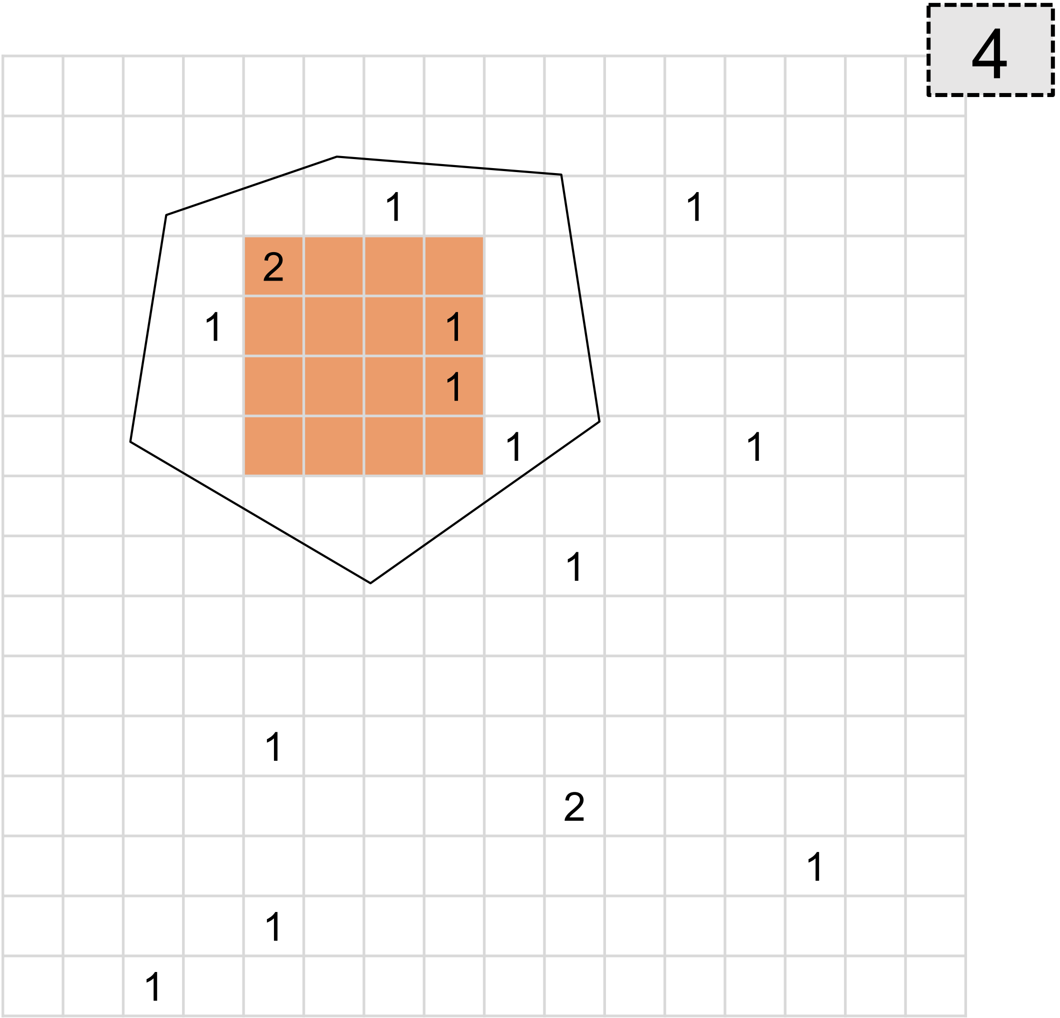
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



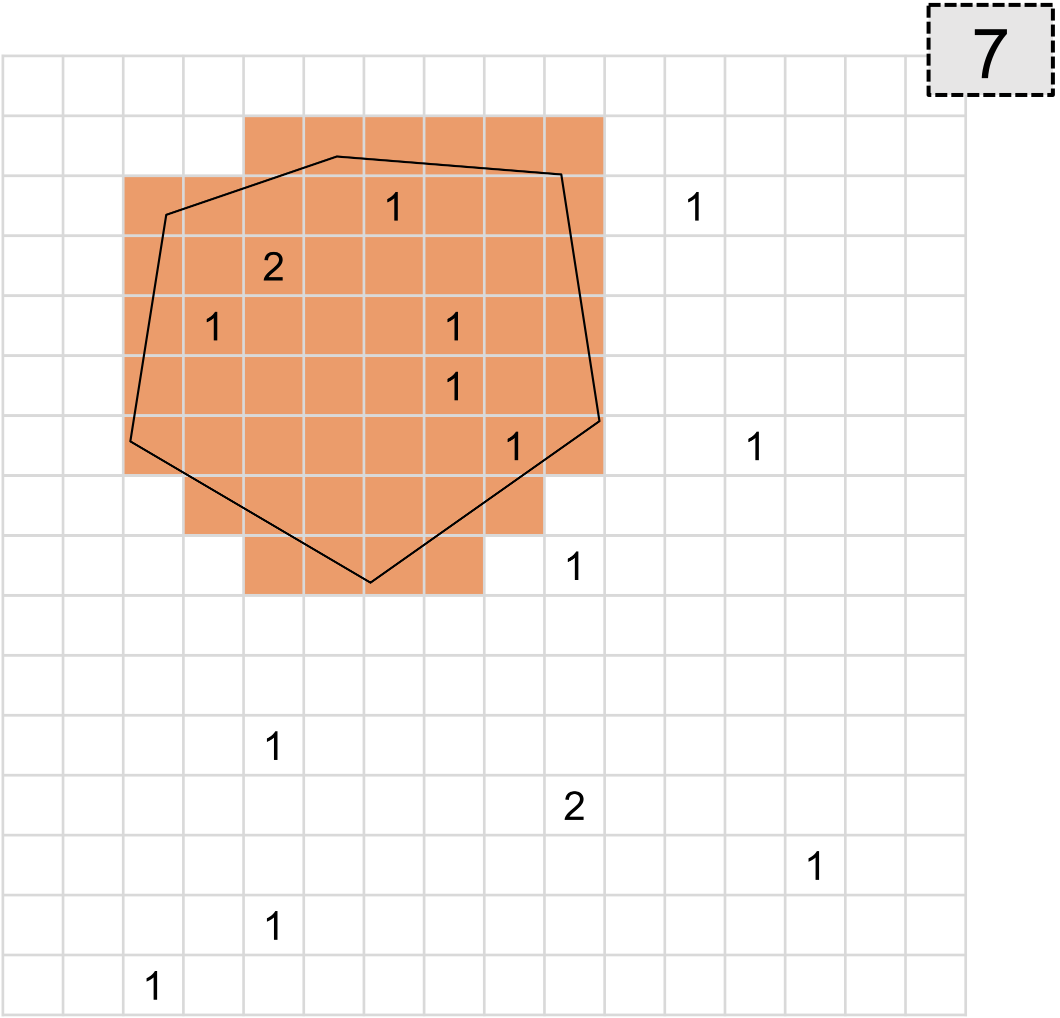
Raster Join

```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



Raster Join

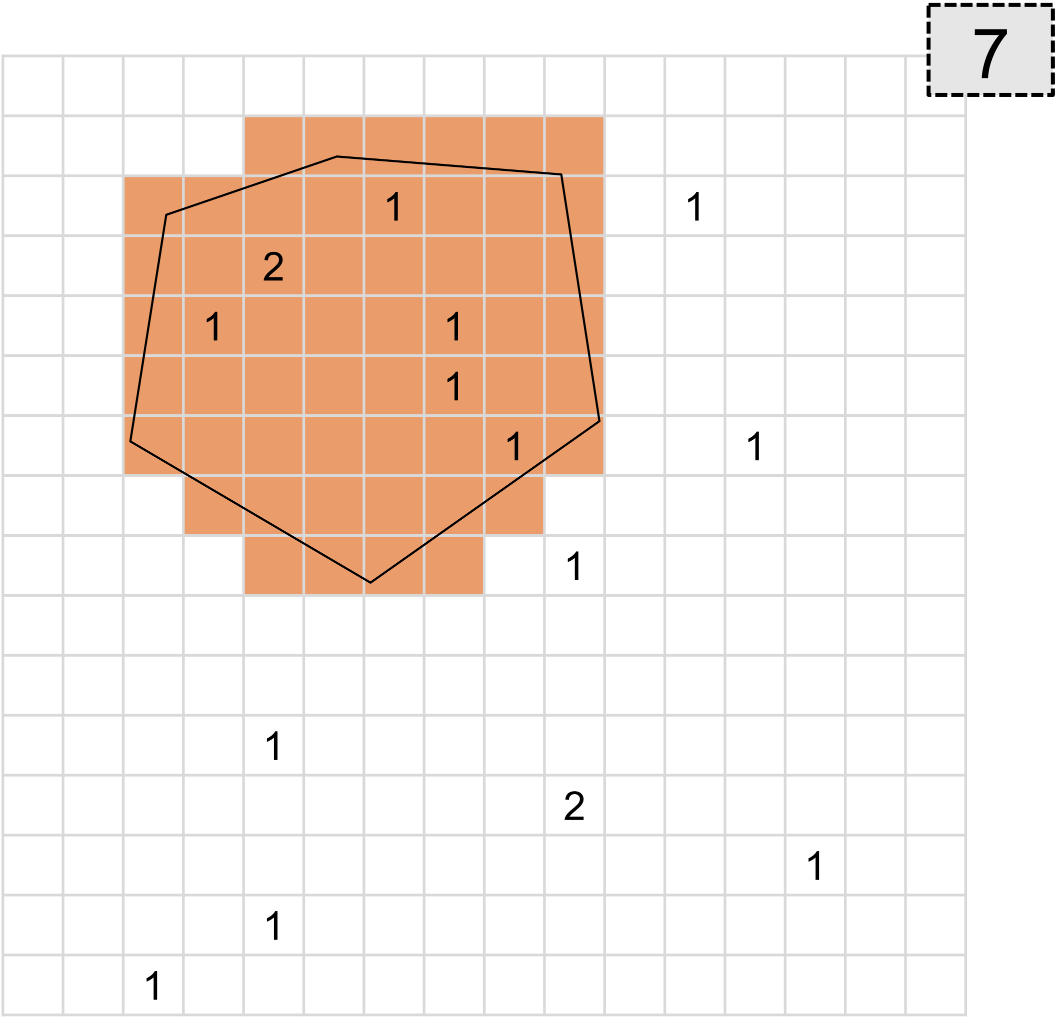
```
SELECT COUNT(*)
FROM taxi T, neighborhoods N
WHERE T.pickup inside N.polygon
GROUP BY N.id
```



UIC COMPUTER SCIENCE

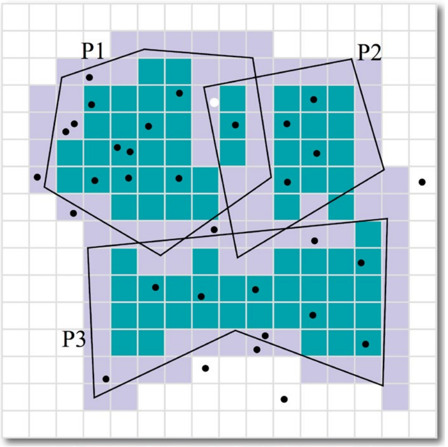
Raster Join

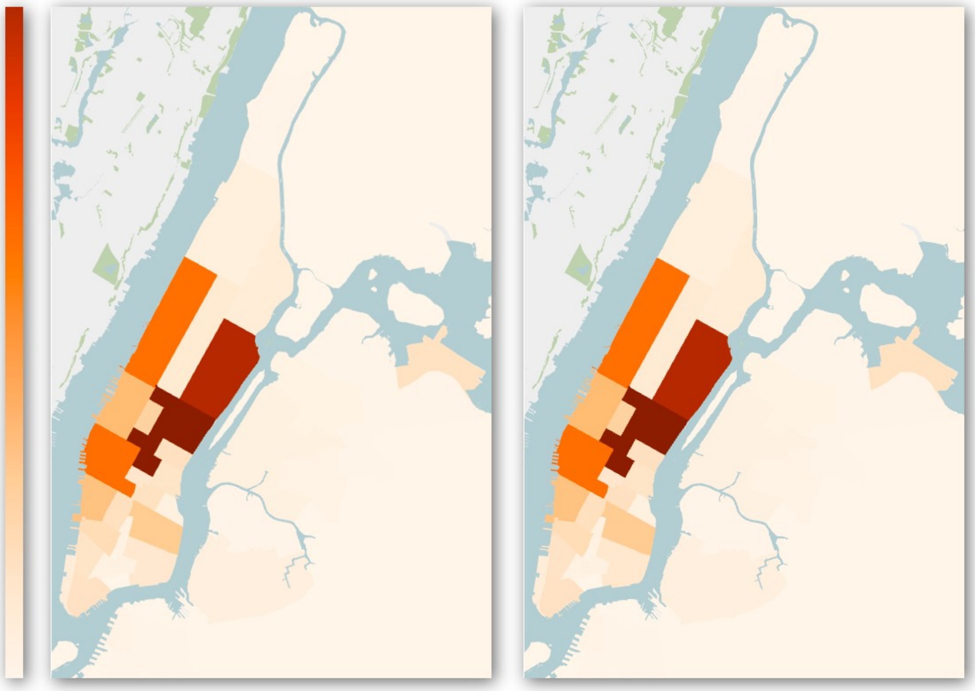
- Exploits native support for drawing in GPUs.
- Combines the aggregation with the join operation.
- No Point-in-Polygon tests.
- Shortcomings?



Raster Join: approximation

- Rasterization can introduce false positives and false negatives.
- Errors can be reduced: approximate the polygon outline by increasing screen resolution (i.e., reducing pixel size).
- Accurate Raster Join: point-in-polygon tests for points in the boundary.



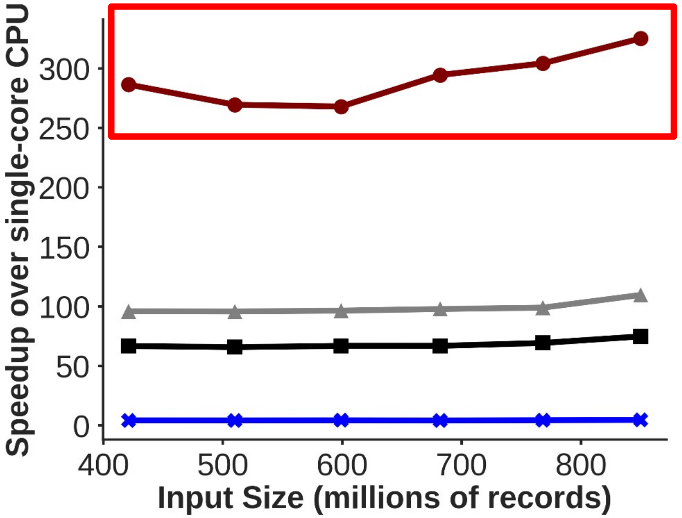


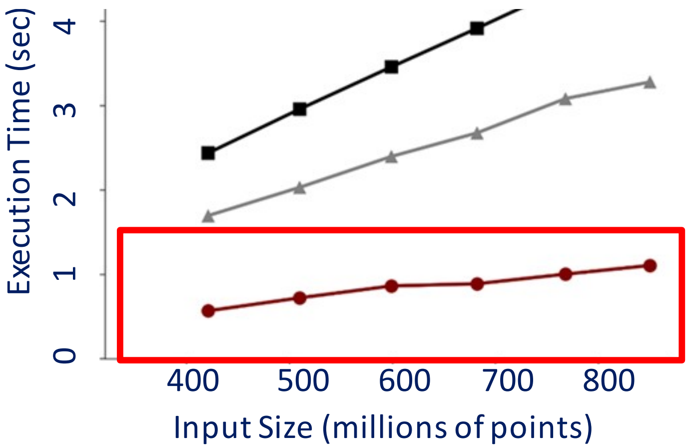
approximate accurate

Raster Join: performance evaluation

NYC taxi data (over 868 million points), 260 NYC neighborhood polygons

Laptop with i7 Quad-Core@2.8 GHz, 16 GB RAM, GTX 1060 GPU





300x speedup