COMP5338 – Advanced Data Models

Week 9: Spatial Data Model and Query

Dr. Ying Zhou
School of Computer Science

Outline

- **Motivation**
- Spatial Data Model
- Spatial Data Queries
- Spatial Query in MongoDB

Motivation

- Many entities represent physical objects, and some physical features also matter for business purposes
  - Eg. A store has a name, business category, contact number and is located at a particular place (geo-spatial)
    - The geo-spatial feature can help find a store that is close to a customer
  - Eg. A toy has name, category, material, and shape
    - The shape feature helps to find a box that can fit the toy
- There are large amount of geospatial data:
  - Businesses and homes have addresses
    - Both the logic aspect and physical aspect
  - Google Maps, Google Earth
  - Weather and Climate Data

Spatial Data and Object Concept

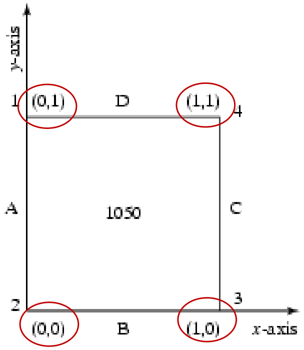
- Spatial feature could refer to a
  - Point, a 2d shape, a 3d shape, or shape in higher dimension
- Spatial features of an entity cannot be represented as simple value type
  - A point in 2d space has two coordinate values
- It is natural to use object to represent spatial features (spatial data)
- There are also spatial related operations we need to perform on spatial data
  - Compute distance between points
  - Compute area of 2d shape or volume of 3d shapes
  - Compute various relationships among spatial objects

Spatial Data as Object Example

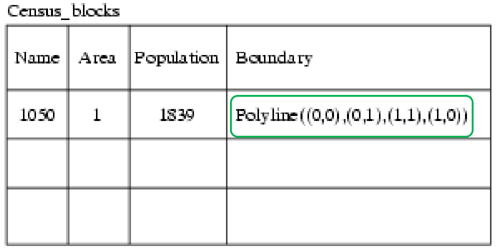
- Consider a spatial data representing census block:

```
CREATE TABLE census_blocks (
   name      string,
   area      float,
   population      number,
   boundary        Polyline );
```

A User Defined Type



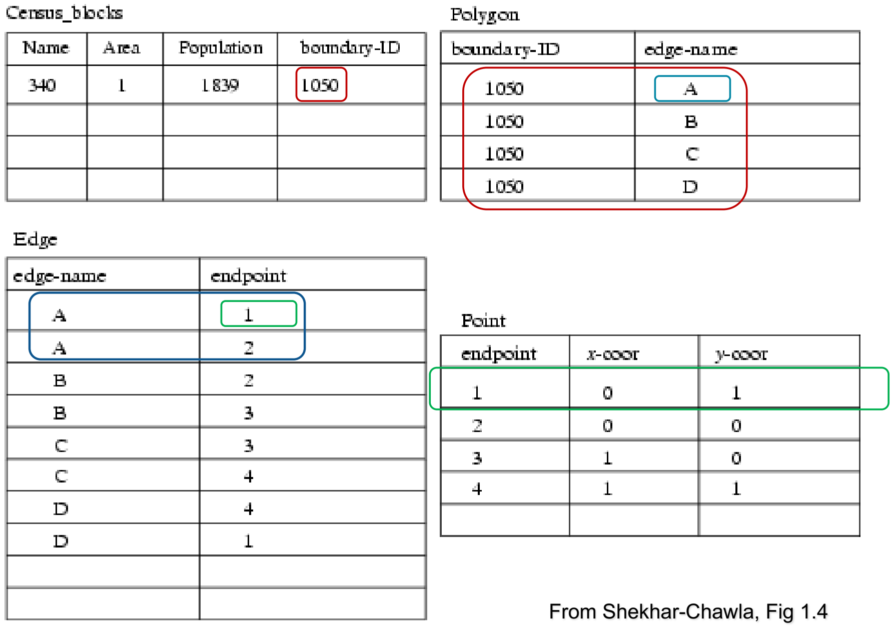
Census_blocks



Name	Area	Population	Boundary
1050	1	1839	Polyline((0,0),(0,1),(1,1),(1,0))

From Shekhar-Chawla, Fig 1.4

Spatial Data in Purely Relational Form



Census_blocks

Name	Area	Population	boundary-ID
340	1	1839	1050

Polygon

boundary-ID	edge-name
1050	A
1050	B
1050	C
1050	D

Edge

edge-name	endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	1

Point

endpoint	x-coor	y-coor
1	0	1
2	0	0
3	1	0
4	1	1

From Shekhar-Chawla, Fig 1.4

Spatial Database Management System

- A SDBMS is a software module that
  - can work with an underlying DBMS
  - supports spatial data models, spatial abstract data types (ADTs) and a query language from which these ADTs are callable
  - supports spatial indexing, efficient algorithms for processing spatial operations, and domain specific rules for query optimization
  - Many RDBMS and NoSQL storage systems have support for spatial data
    - Oracle, SQL Server, MongDB, Neo4j, …
- SDBMS components
  - spatial data model
  - query language
  - query processing
  - file organization and indices
  - query optimization
  - etc.

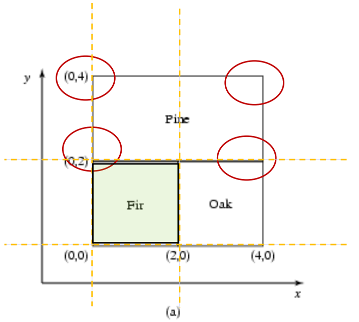
Outline

- Motivation
- **Spatial Data Model**
  - **Field vs. Object Models**
  - **Coordinate System**
  - **Topological Operations**
- Spatial Data Queries
- Spatial Query in MongoDB

Models of Spatial Information

- Two common models
  - Field based *(also: space-based)*
    - Model properties of underlying space
    - Good for expressing values vary continuously over space (e.g. temperature, rainfall, elevation, depth, etc.
    - Fields are actually **functions** that map spatial locations to values
  - Object based *(also: entity-based)*
    - Model boundaries of spatial feature (e.g. the census block is modelled by a polygon)
    - Good for expressing discrete spatial entities

Examples of Field and Object Models



(a) forest stand map
(b) Object model has 3 polygons defining the boundaries
(c) Field model uses a function to calculate the property "tree species"

Object Viewpoint of Forest Stands

Area-ID	Dominant Tree Species	Area/Boundary
FS1	Pine	[(0,2),(4,2),(4,4),(0,4)]
FS2	Fir	[(0,0),(2,0),(2,2),(0,2)]
FS3	Oak	[(2,0),(4,0),(4,2),(2,2)]

(b)

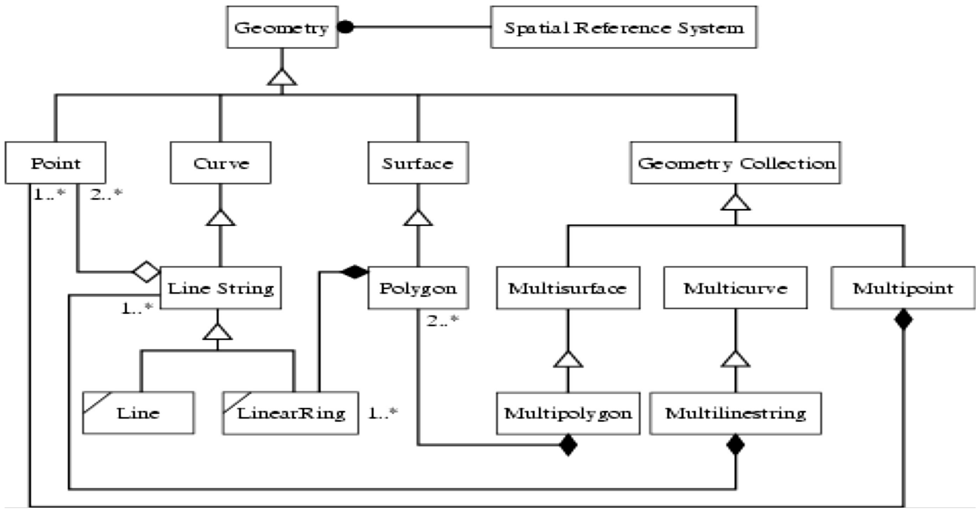
Field Viewpoint of Forest Stands

$$f(x,y) = \begin{cases} \text{"Pine,"} & 2 \leqslant x \leqslant 4 ; 2 < y \leqslant 4 \\ \text{"Fir,"} & 0 \leqslant x \leqslant 2; 0 \leqslant y \leqslant 2 \\ \text{"Oak,"} & 2 < x \leqslant 4; 0 \leqslant y \leqslant 2 \end{cases}$$

(c)

From Shekhar-Chawla, Fig 2.1

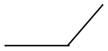
OpenGIS Geometry Model



From Shekhar-Chawla, Fig 2.2

Example Spatial Objects

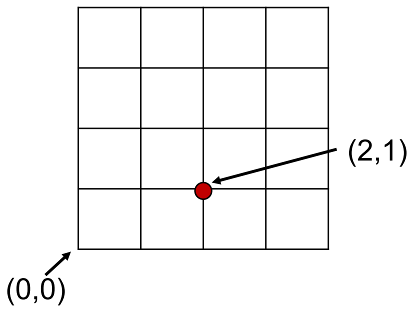
- **Point**: represented by its coordinates eg (-10, 30)
- Collection of several **points**
- **Line String**
  - Simplest form is piece-wise linear, given by points (and implying the straight segments between them)
  - Eg (0,1), (1,1), (2,2)



- **Polygon**
  - A 2-d region whose boundary is given
  - Simplest form: boundary is a **line string** that returns to its start
  - More complicated: region with holes
- Collection of **polygons**

Coordinate Systems

- Points from a 2-d space are represented by pairs of numbers
  - The numbers could refer to a dot on a drawing area, a piece of land in a game setting, or a location on earth
- There are many ways to associate numbers with points
- Simple 2d Cartesian coordinate
  - Choose a point as origin (0,0)
  - Choose a direction for the x-axis, and a scale (how far is (1,0)) from (0,0)?)



A Round World

- The surface of the earth is 2-dimensional, but curved
  - Cartesian systems work reasonably in small regions
- Traditional geographic coordinate system
  - 2d: longitude and latitude
  - 3d: longitude, latitude, elevation
  - The surface is a sphere
    - (-179,10) is very close to (179, 10)
    - A linestring might cross the (long = 180) line;
- Many SDBMS supports both flat space and sphere

Operations on Spatial Objects in the Object Model

- Classifying operations
  - ***Set based***:
    - a set operation (e.g. intersection) of 2 polygons produce another polygon
  - ***Topological operations***: Boundary of USA touches boundary of Canada
  - ***Directional***: New York city is to east of Chicago
  - ***Metric***: Chicago is about 700 miles from New York city.

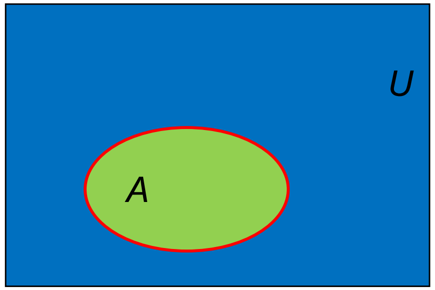
Set theory based	Union, Intersection, Containment
Topological	**Touches, Disjoint, Overlap, etc.**
Directional	East,North-West, etc.
Metric	Distance

Topological Relationships

- Topological Relationships
  - invariant under elastic deformation (without tear, merge).
  - Two countries which touch each other in a planar paper map will continue to do so in spherical globe maps.
- Example queries with topological operations
  - What is the topological relationship between two objects A and B ?
  - Find all objects which have a given topological relationship to object A?
    - E.g. find all rivers that cross a city
- Can we express *topological relationship* mathematically?
  - Metric operations may be expressed using various functions, e.g. distance function
  - Set operations can be express mathematically
  - The mathematical form helps to calculate such relationships

Topological Concepts

- Interior, boundary, exterior
  - Let *A* be an object in a “Universe” *U*.



Green is *A* interior (A^o)

Red is boundary of *A* (∂A)

Blue – (Green + Red) is *A* exterior (A^-)

- Exterior is also referred to as the *complement* of an object

Nine-Intersection Model of Topological Relationships

- Many topological Relationship between A and B can be specified using 9 intersection model
  - Eight possible 2D topological relationships for objects without holes;
- Nine intersections
  - intersections between interior, boundary, exterior of A, B
  - A and B are spatial objects in a two dimensional plane.
  - Can be arranged as a 3 by 3 matrix
  - Matrix element take a value of 0 (false) or 1 (true)

$$\Gamma_9(A,B) = \begin{pmatrix} A^0 \cap B^0 & A^0 \cap \partial B & A^0 \cap B^- \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^0 & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

From Shekhar-Chawla, p 28

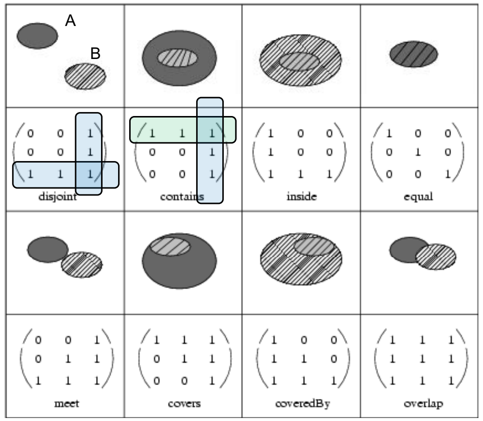
Specifying Topological Operations using the 9-Intersection Model

B

		I	B	E
A	I			
	B			
	E			

For **disjoint** relation:
A’s exterior intersects with B’s everything and vice versa

If A **contains** B:
A’s interior intersects with B’s everything and B’s exterior intersects with A’s everything



From Shekhar-Chawla, Fig 2.3

Outline

- Motivation
- Spatial Data Model concepts
- **Spatial Data Queries**
  - **Query type**
  - **General processing steps**
- Spatial Query in MongoDB

Spatial Processing

- Find one or more entities, based on *non-spatial* aspects, then use *spatial operations* to get interesting data associated with these items
  - Eg find the *length* of the river called '*Mississipi*'
    - Find the river based on name (*non-spatial*), use *spatial operation* to compute the length (assuming it is not stored as a numeric value)
  - Eg find the total area of all counties whose population exceeds 1,000,000
    - Find the counties with population exceeds 1,000,000 (*non-spatial*), compute each county's area(*spatial operation*) and sum all up.
- To answer these: use conventional index or table scan to find the appropriate rows, then call spatial functions on the spatial attribute of each

Spatial selection queries

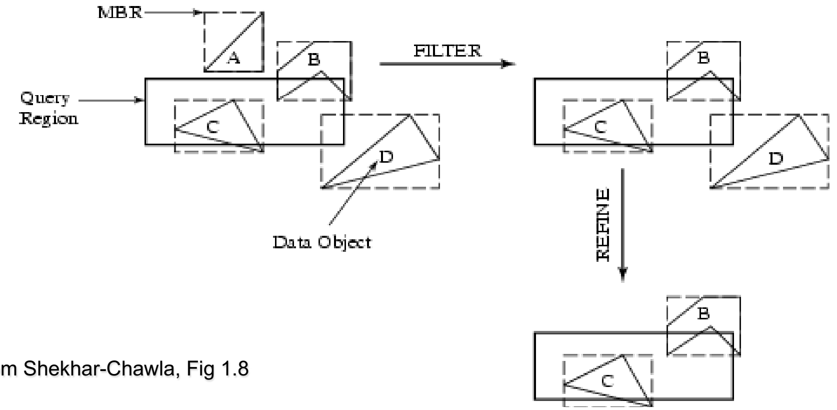
- Find items whose spatial attribute has certain properties
  - Eg find rivers whose length is at least 10000
- `WHERE` clause will involve spatial operations
- Typical processing: scan all rows, apply appropriate spatial operation to the spatial attribute of each

Spatial Range Query

- A particular kind of spatial selection, in which the condition involves a topological or metric relationship to a given object
  - Eg find all bookshops whose location is inside a given region
  - Eg find all farms that contain (part of) a given curve
  - Eg find all rivers that flow through a given region
  - Eg find all bookshops within 100 km of a given point
    - Equivalent to: find all whose location is inside a circular region of radius 100 km!
- Simple processing: scan the appropriate table, apply appropriate spatial operation to each item's spatial attribute
- But often one can do better: *first filter* to find a small set where the condition might be feasible; then *refine* the list by checking in detail each that pass the filter

Spatial Query Processing: Filter-Refine Strategy

- Eg find objects that intersect a query region
- **Filter Step**: made easy if each object has associated to it a simple shape that surrounds it (eg Minimum Bounding Rectangle)
  - If object's MBR doesn't intersect query {or MBR of query}, there is no possibility that the object itself will intersect the query
- **Refine Step**: Actually perform intersection method for those objects that get through the filter



From Shekhar-Chawla, Fig 1.8

Nearest neighbours

- Find entities that are as close as possible to given location, Always give a bound on how many to find (K Nearest Neighbours)
  - Eg find 5 closest restaurants to (100, 350) and return them ranked by closeness
- Simple processing: scan all, compute distance; keep track of the ones with lowest distances seen so far
- Many index based algorithms, see next week

Spatial Join Query

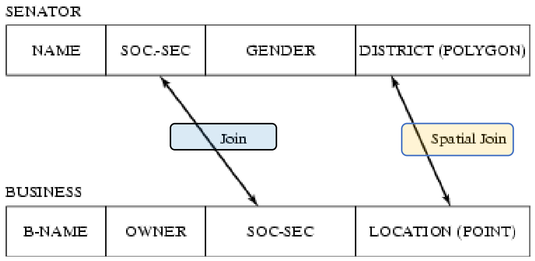
- Spatial join example

```
SELECT      S.name
  FROM      Senator S, Business B
  WHERE     S.district.Area() > 300
            AND Within(B.location, S.district)
```

- Non-Spatial Join example

```
SELECT S.name
     FROM    Senator S, Business B
     WHERE   S.soc-sec = B.soc-sec AND S.gender = 'Female'
```

- Similar to non-spatial join, spatial join are usually very expensive to process



From Shekhar-Chawla, Fig 1.7

Outline

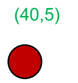
- Motivation
- Spatial Data Model concepts
- Spatial Data Queries
- **Spatial Query in MongoDB**

Spatial data: GeoJSON

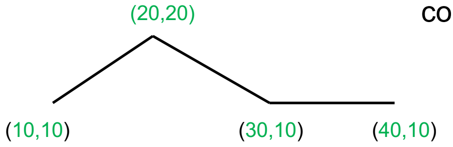
- Spatial data in MongoDB can be stored as **`GeoJSON`** object or as legacy coordinate pairs
  - **`GeoJSON`** data assumes *earth-like* sphere
  - Legacy coordinate pairs assumes flat plane
- **`GeoJSON`** object uses JSON format to represent spatial objects in OpenGIS
  - Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, Geometry Collection
  - General format

    { type: "<GeoJSON type>" , coordinates: <coordinates>}
  - The coordinate reference system for all GeoJSON coordinates is a geographic coordinate reference system, using the World Geodetic System 1984 (WGS 84) [WGS84] datum, with longitude and latitude units of decimal degrees.

Point and LineString

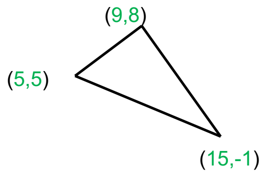


{type: “Point”, coordinates: [40,5]}



{type: “LineString”,
coordinates: [
[10,10], [20,20], [30,10], [40,10]] }

A closed LineString is called a linear ring, with at least four coordinate pairs and specify the same position as the first and last coordinates.

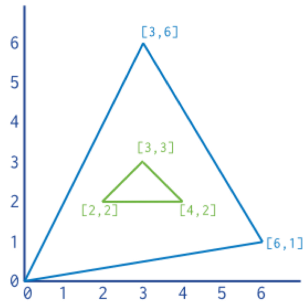


{type: “LineString”,
coordinates: [**[5,5]** ,[9,8], [15,-1], **[5,5]**]
}

Polygon

- Polygon is used to model two dimensional surface
  - Triangle, Rectangle, Pentagon, …
  - A polygon is represented as one or many linear rings, the first is the exterior ring bounds the surface, the others are interior rings bound holes within the surface

```
{
  type : "Polygon",
  coordinates : [
    [ [ 0 , 0 ] , [ 3 , 6 ] , [ 6 , 1 ] , [ 0 , 0 ] ],
    [ [ 2 , 2 ] , [ 3 , 3 ] , [ 4 , 2 ] , [ 2 , 2 ] ]
  ]
}
```



Model multiple disjoint objects

- MultiPoints, MultiPolygon

```
{ type: "MultiPoint",
    coordinates: [ [ -73.9580, 40.8003 ],
                   [ -73.9498, 40.7968 ],
                   [ -73.9737, 40.7648 ],
                   [ -73.9814, 40.7681 ] ] }

{ type: "MultiPolygon",
    coordinates: [ [ [ [ -73.95, 40.80 ], [ -73.9498, 40.79 ], [ -73.97, 40.76 ], [ -73.95, 40.80 ] ] ],
                   [ [ [ -73.95, 40.80 ], [ -73.94, 40.79 ], [ -73.97, 40.76 ], [ -73.95, 40.80 ] ] ] ] }
```

- Geometry collection

```
{ type: "GeometryCollection",
  geometries: [ { type: "MultiPoint",
                  coordinates: [ [ -73.95, 40.80 ], [ -73.94, 40.79 ]]},
                { type: "MultiLineString",
                  coordinates: [… ] }
              ] }
```

Spatial Queries

- `$near` and `$nearSphere`
  - Specifies a point for which a geospatial query returns the documents from nearest to farthest
  - Can be used to run queries like find all car parks/restaurants within certain distance
- `$geoWithin`
  - Find all geo objects contained in a query shape
- `$geoIntersects`
  - Find all geo objects intersects with a query shape. Here intersect includes relationships such as cover, equal, overlap, touch and so on.
- Others

Spatial Index

- **2dsphere** indexes supports all MongoDB geospatial queries
- Eg.

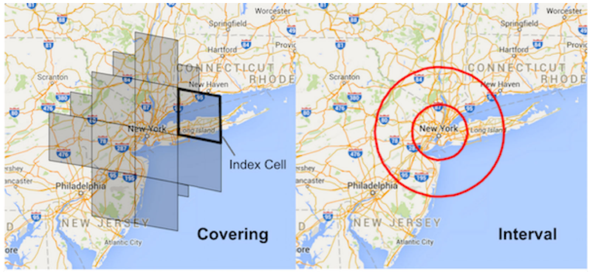
```
db.places.insert(
  {
    loc : { type: "Point", coordinates: [ -73.97, 40.77 ] },
    name: "Central Park",
    category : "Parks"
  }
)
```

- db.places.createIndex({ loc : "2dsphere" })

MongoDB Spatial Index

- " MongoDB's 2dsphere index actually combines the strength of discrete global grids and B+ -tree structures, which first partitions the Earth surface into cells at multiple resolution levels and then applies a B+ -tree to index geographical features approximated as one or multiple cells."

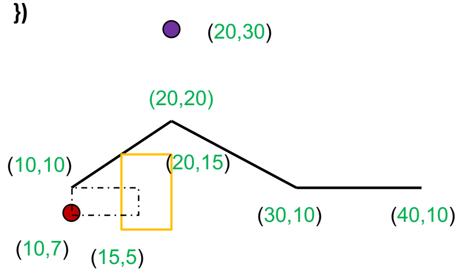
L. Xiang, J. Huang , X. Shao and D. Wang: MongoDB-Based Management of Planar Spatial Data with a Flattened R-Tree
https://pdfs.semanticscholar.org/860f/0cfe3e4b4cb66b2b2895016a60e824e9e9f8.pdf



MongoDB Blog: Geospatial Performance Improvements in MongoDB 3.2
https://www.mongodb.com/blog/post/geospatial-performance-improvements-in-mongodb-3-2

Spatial Query -- $geoIntersects

```
db.places.find({
     loc :{$geoIntersects:
               {$geometry:{
                         type: ”Polygon”,
                         coordinates:[[[10,7], [10,10], [17,10],[17,7],[10,7]]]}
               }
          }
})
```



References

- S. Shekhar and S.Chawla: *Spatial Databases: A Tour*. Prentice Hall, 2002. [http://www.spatial.cs.umn.edu/Book/]
  - Chapter 1-3
- MongoDB document on geospatial query
  - https://docs.mongodb.com/manual/geospatial-queries/