# Spatial DBs

# Objectives/TOC

- spatial DBs: definition, characteristics, need, creation..
- spatial datatypes
- spatial operators
- spatial indices
- implementations
- miscellany

# What is a spatial database?

"A spatial database is a database that is optimized to store and query data related to objects in space, including points, lines and polygons."

In other words, it includes objects that have a SPATIAL location (and extent). A chief category of spatial data is geospatial data - derived from the geography of our earth.

Characteristics of geographic data:

- has location
- has size
- is auto-correlated
- scale dependent
- might be temporally dependent too

Geographic data is NOT 'business as usual'!

---

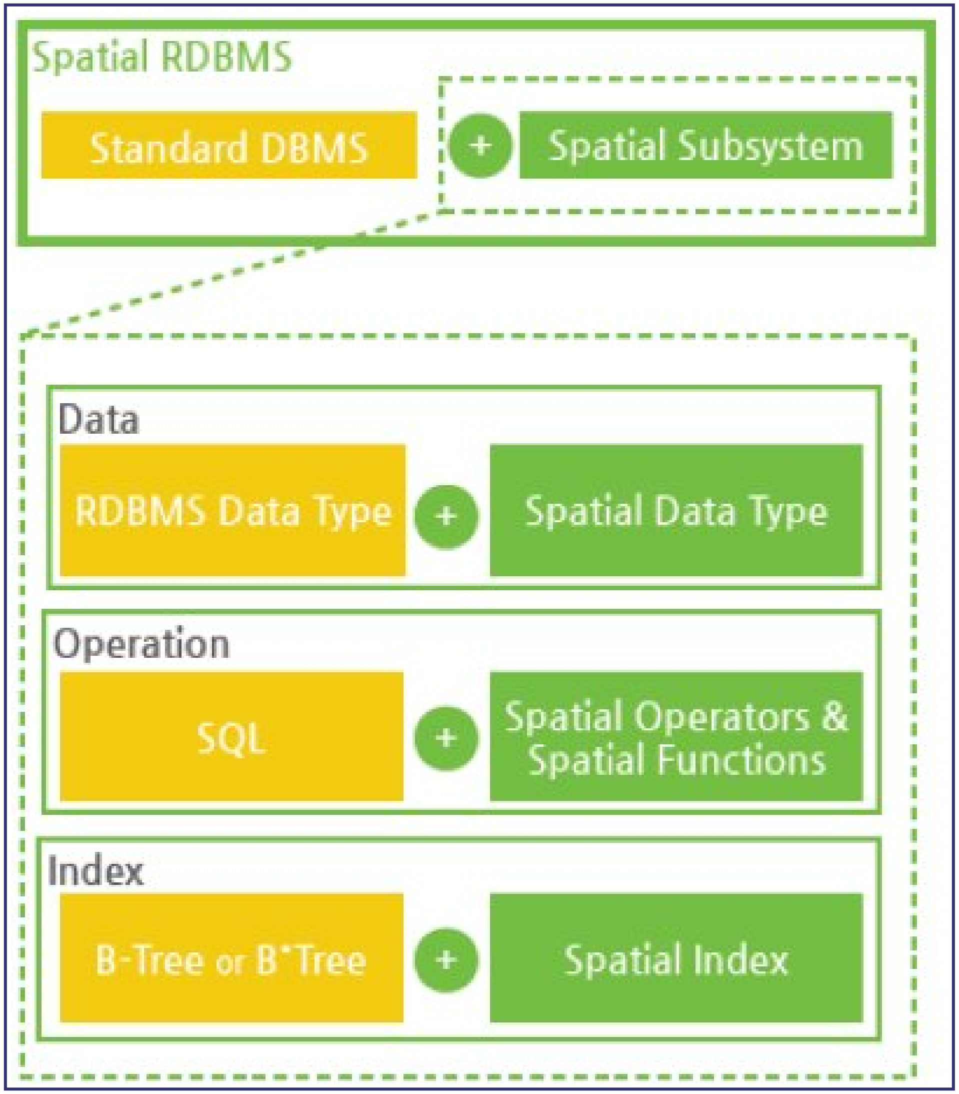
## Entity view vs field view

In spatial data analysis, we distinguish between two conceptions of space:

- entity view: space as an area filled with a set of discrete objects
- field view: space as an area covered with essentially continuous surfaces

For our purposes, we will adopt the 'entity' view, where space is populated by discrete objects (roads, buildings, rivers..).

# Components




So a spatial DB is a collection of the following, specifically built to handle spatial data:

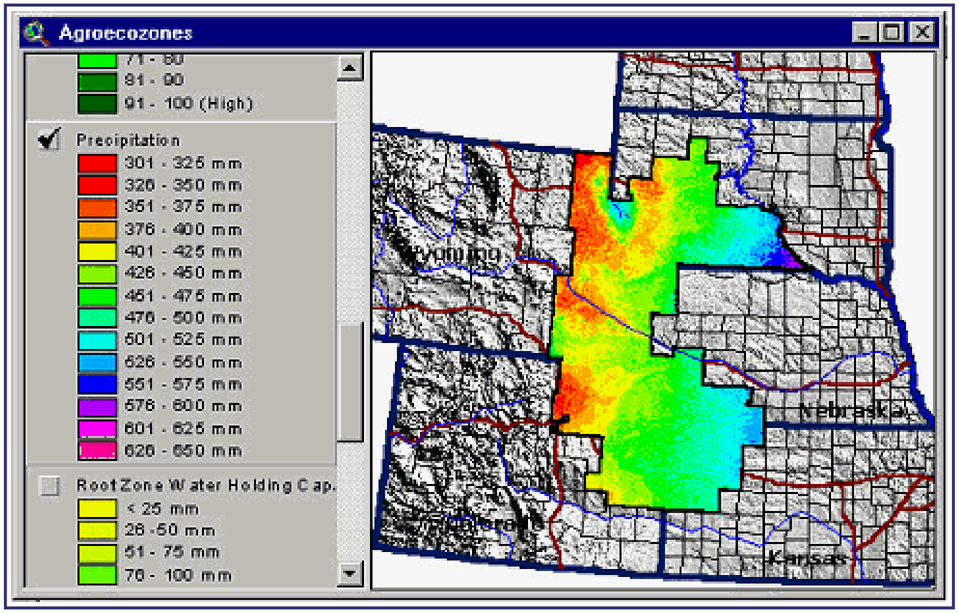
- types
- operators
- indices

Soon, we will explore what types, operators and indices mean.

# Examples of spatial data

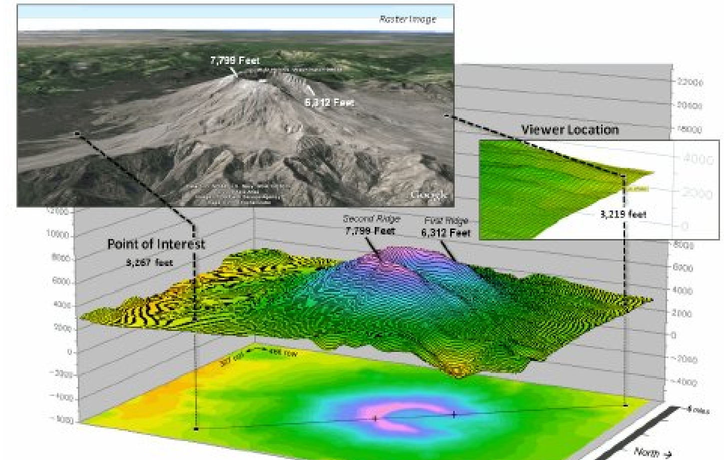
CAD data:


Agricultural data:



3D data:

- Three-dimensional data examples
  - Weather
  - Cartesian coordinates (3-D)
  - Topological
  - Satellite images




Digital Elevation Model (DEM)
(Grid Surface configured as 466 rows by 327 columns with 30m cells )

# What can be plotted on to a map?

- crime data
- spread of disease, risk of disease [look at this too]
- drug overdoses - over time
- census data
- income distribution, home prices
- locations of Starbucks (!)
- (real-time) traffic
- agricultural land use, deforestation

---

## Who creates/uses spatial data?

- **Army Field Commander**: Has there been any significant enemy troop movement since last night?
- **Insurance Risk Manager**: Which homes are most likely to be affected in the next great flood on the Mississippi?
- **Medical Doctor**: Based on this patient's MRI, have we treated somebody with a similar condition ?
- **Molecular Biologist**:Is the topology of the amino acid biosynthesis gene in the genome found in any other sequence feature map in the database ?
- **Astronomer**:Find all blue galaxies within 2 arcmin of quasars.

Various government agencies routinely coordinate spatial data collection and use, operating in effect, a national spatial data infrastructure (NSDI) - these include federal, state and local agencies. At the federal level, participating agencies include:

- Department of Commerce
  - Bureau of the Census
  - NIST
  - NOAA
- Department of Defense
  - Army Corps of Engineers
  - Defense Mapping Agency
- Department of the Interior
  - Bureau of Land Management

- Fish and Wildlife Service
- U.S Geological Survey [earthquakes, map projections]

- Department of Agriculture
  - Agricultural Stabilization and Conservation Service
  - Economic Research Service
  - Forest Service
  - National Agriculture Statistical Service
  - Soil Conservation Service
- Department of Transportation
  - Federal Highway Administration
- Environmental Protection Agency
- NASA

As you can see, spatial data is a SERIOUS resource, vital to US' national interests.

# Where does spatial data come from?

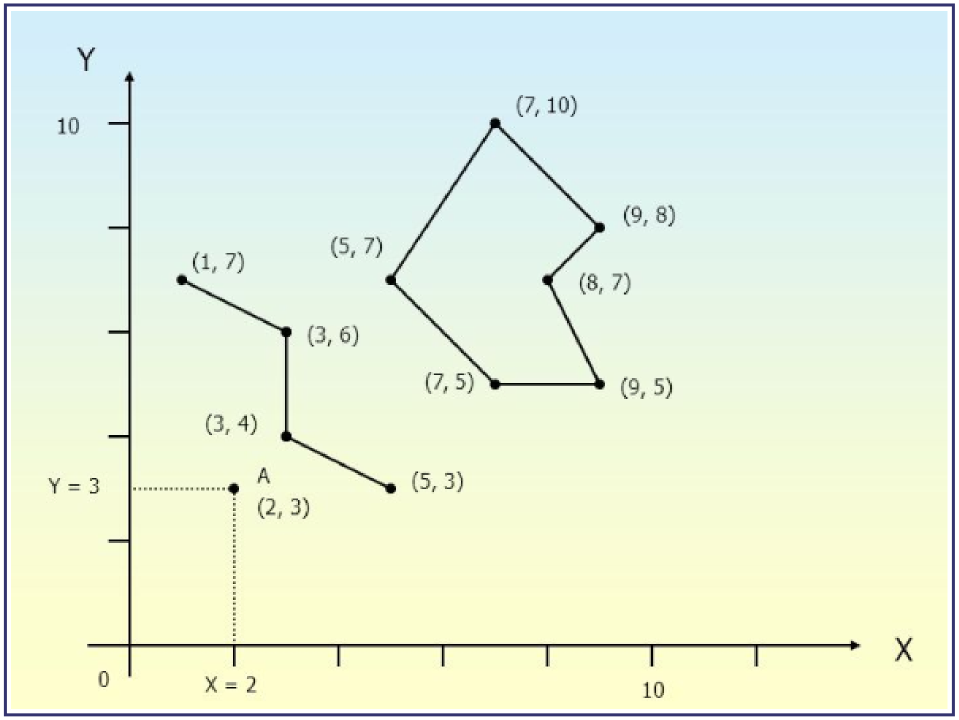
Spatial data is created in a variety of ways:

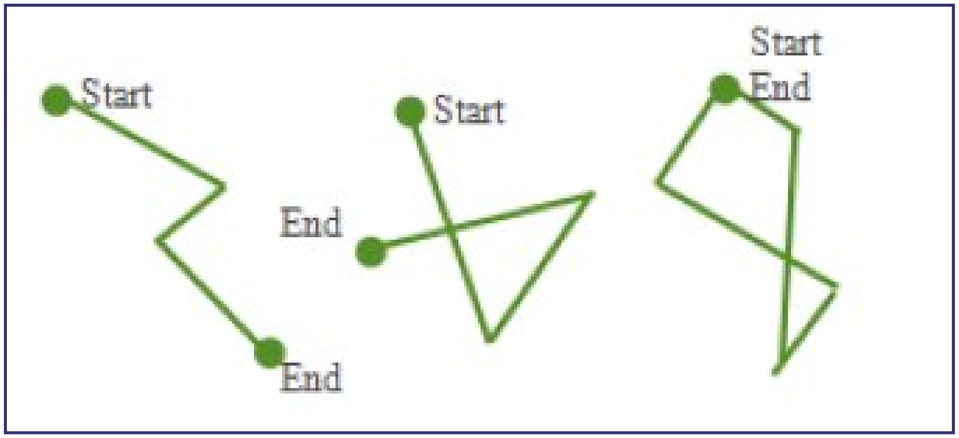
- CAD: user creation
- CAD: reverse engineering
- maps: cartography (surveying, plotting)
- maps: satellite imagery
- maps: 'copter, drone imagery
- maps: driving around
- maps: walking around

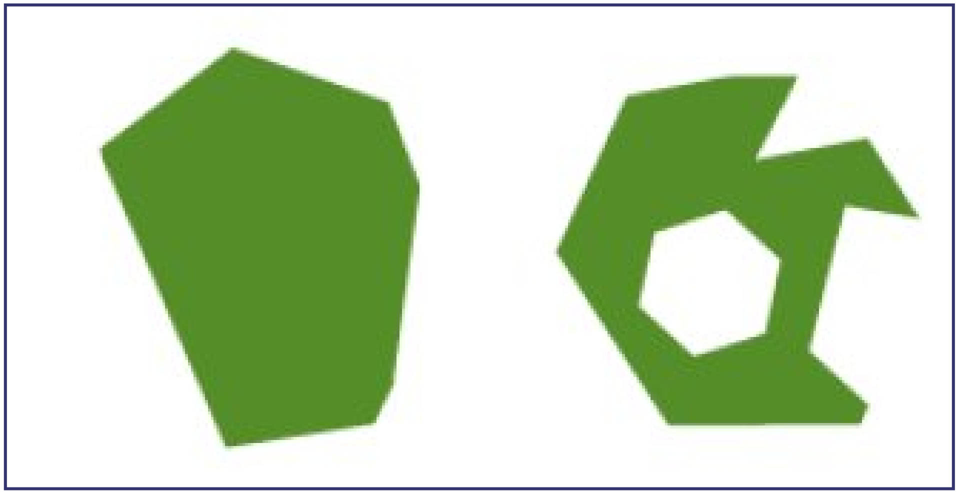
# What to store?

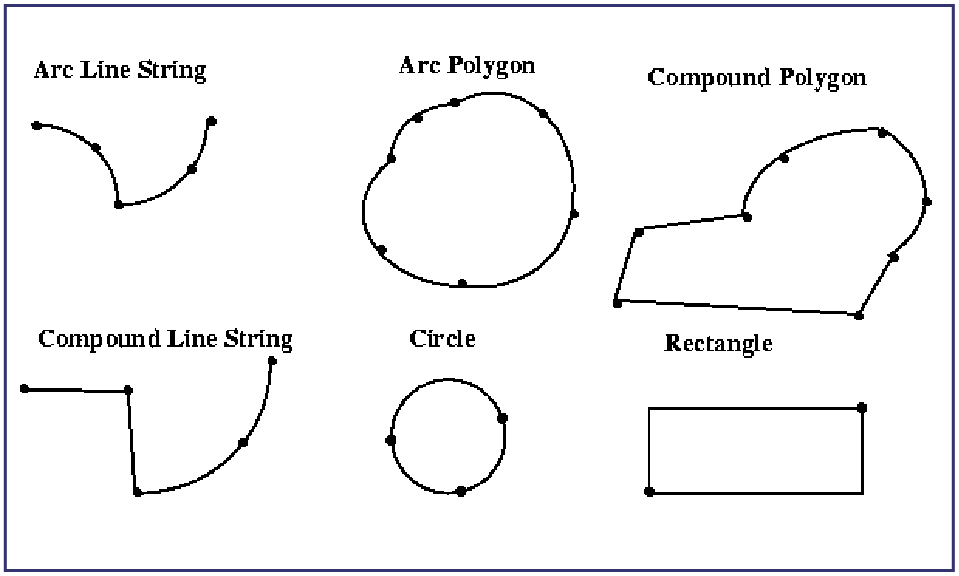
All spatial data can be described via the following entities/types:

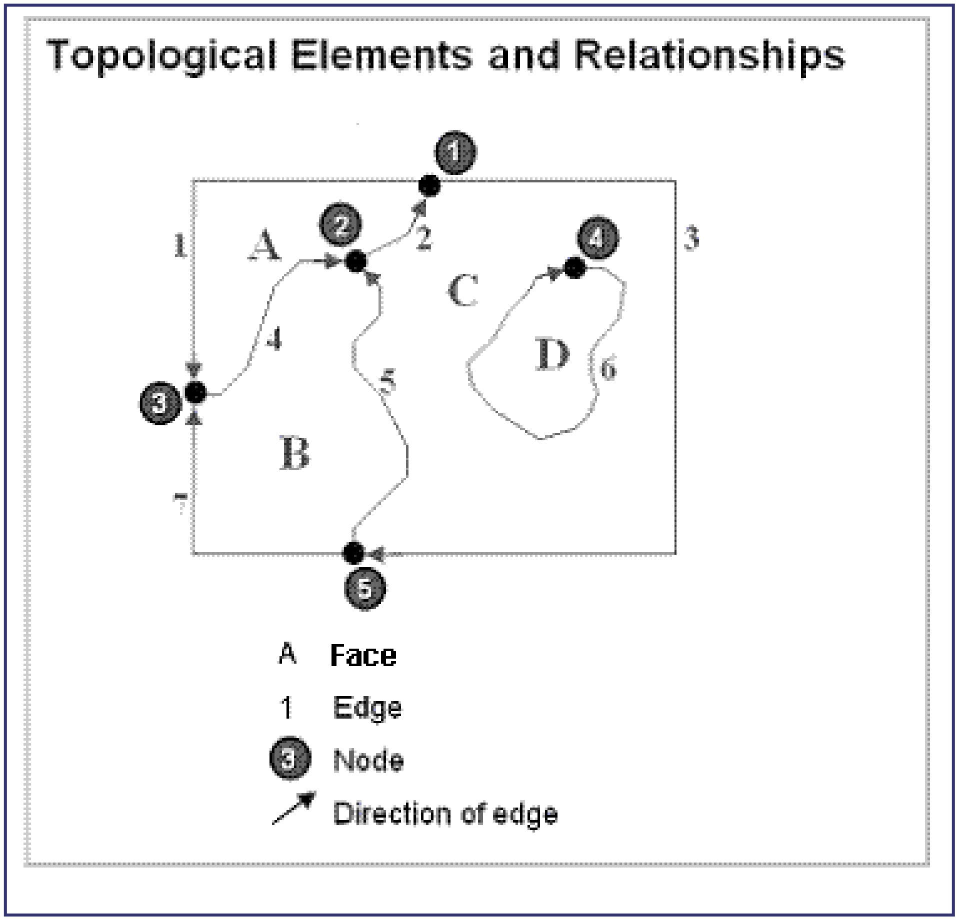
- points/vertices/nodes
- polylines/arcs/linestrings
- polygons/regions
- pixels/raster

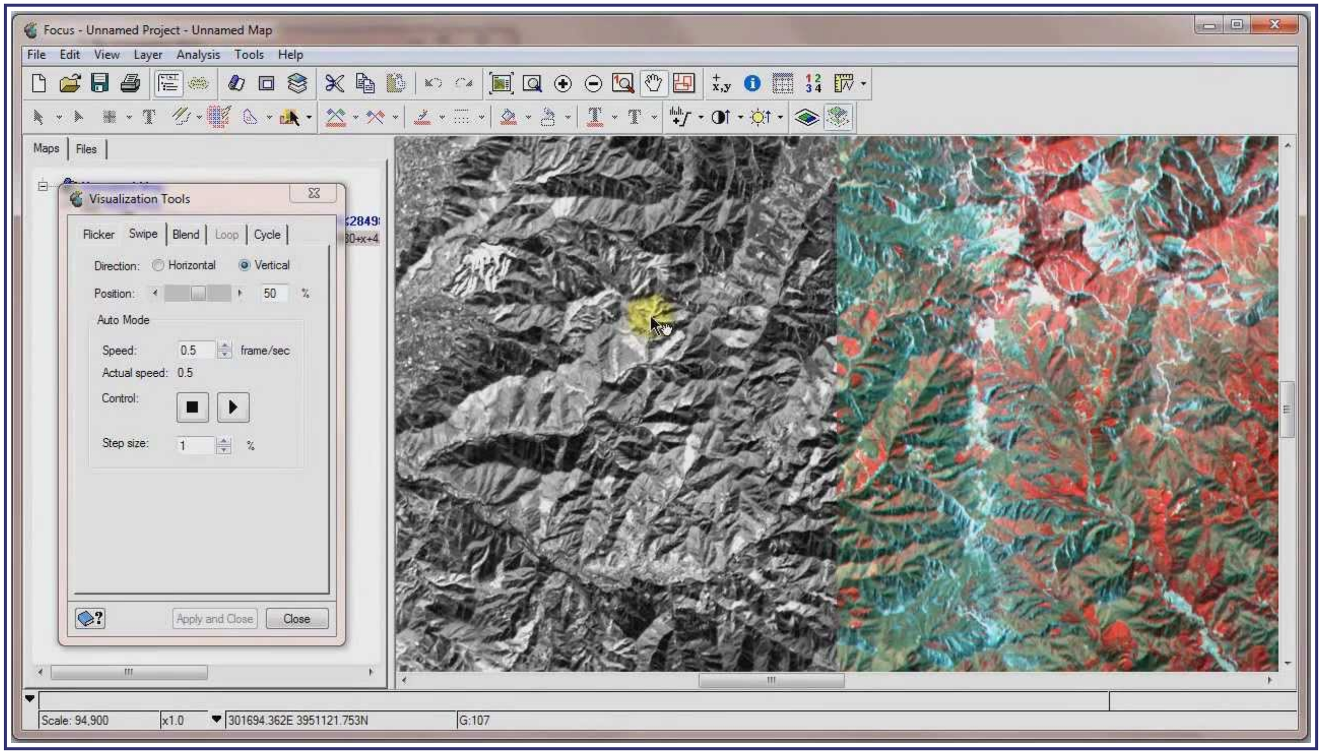



Start
End
Start
End
Start
End




Arc Line String
Arc Polygon
Compound Polygon
Compound Line String
Circle
Rectangle

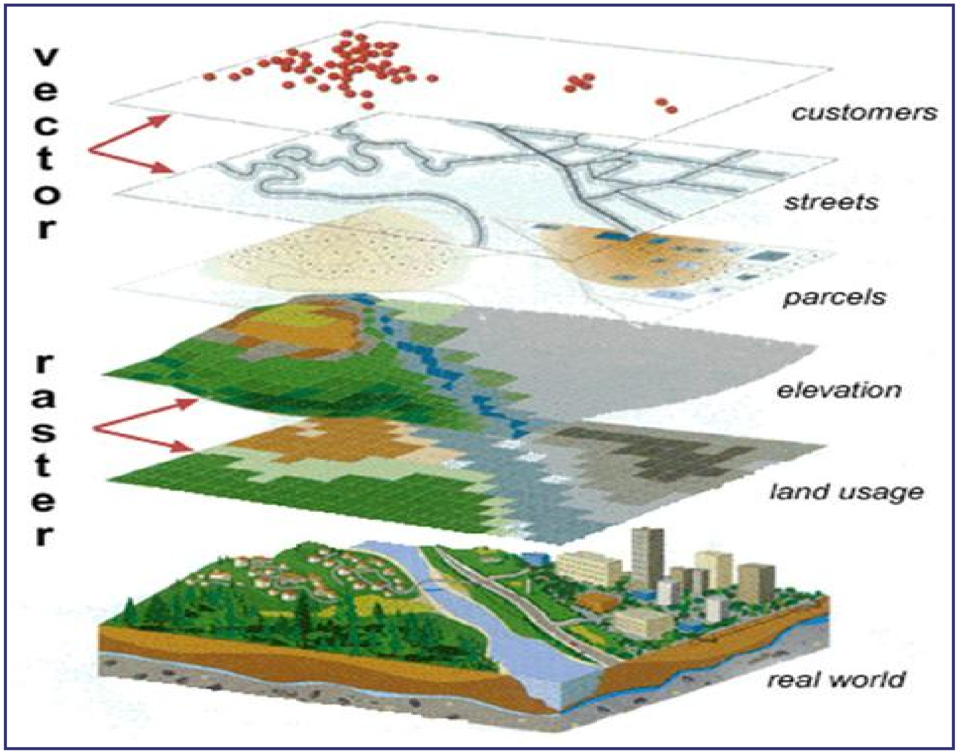

Topological Elements and Relationships
1
A
2
2
1
4
3
C
D
4
5
6
3
B
7
5
A Face
1 Edge
3 Node
Direction of edge

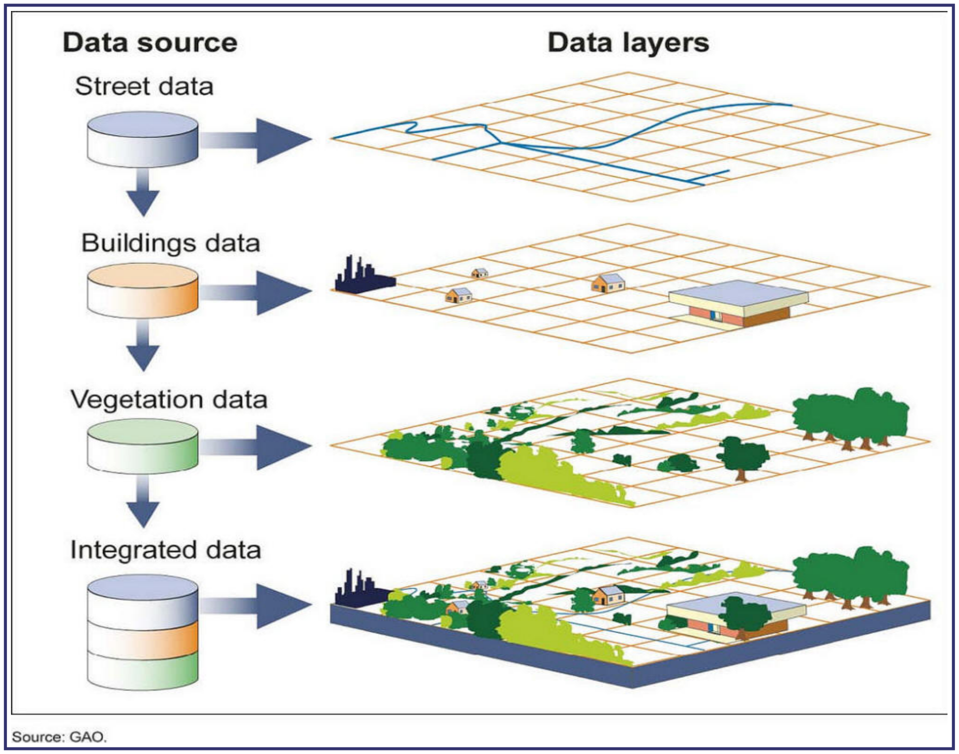

Focus - Unnamed Project - Unnamed Map
File Edit View Layer Analysis Tools Help
Maps Files
Visualization Tools
Flicker Swipe Blend Loop Cycle
Direction: Horizontal Vertical
Position: 50 %
Auto Mode
Speed: 0.5 frame/sec
Actual speed: 0.5
Control:
Step size: 1 %
Apply and Close Close
Scale: 94,900 x1.0 301694.362E 3951121.753N G:107

# Points, lines, polys => models and non-spatial attrs

Once we have spatial data (points, lines, polygons), we can:

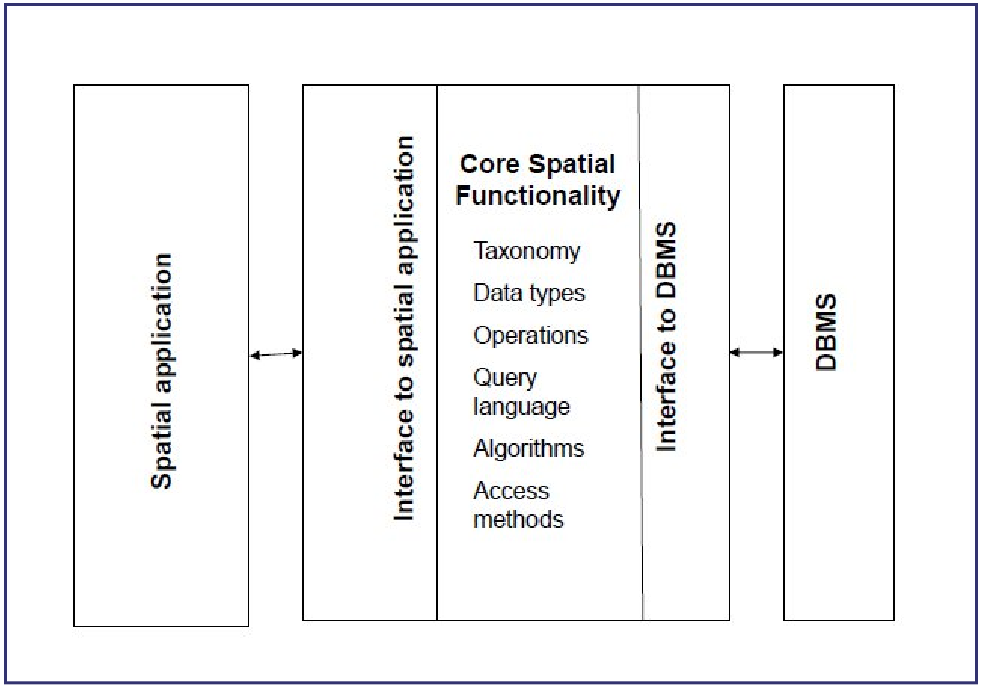
- 'model' features such as lakes, soil type, highways, buildings etc, using the geometric primitives as underlying types
- add 'extra', non-spatial attributes/features to the underlying spatial data

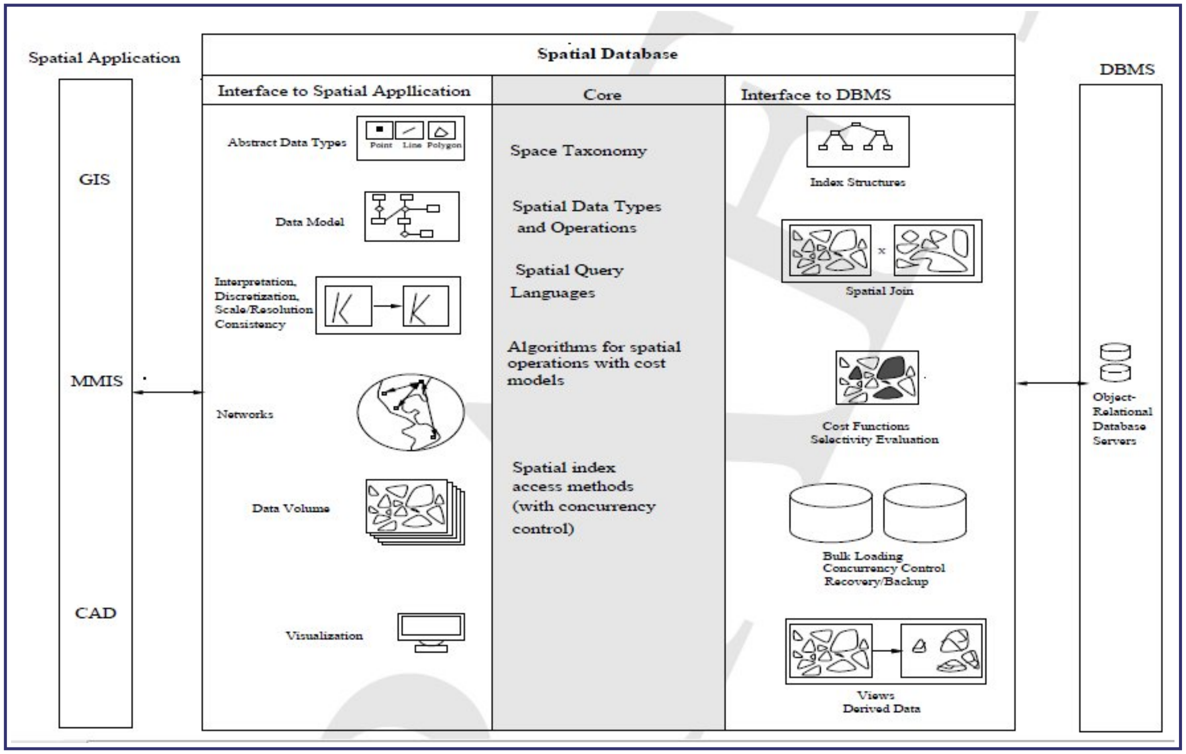





Look at this map, overlaid with scary data..

# SDBMS architecture




Spatial Application
GIS
MMIS
CAD
Spatial Database
Interface to Spatial Appllication
Abstract Data Types
Point
Line
Polygon
Data Model
Interpretation, Discretization, Scale/Resolution Consistency
Networks
Data Volume
Visualization
Core
Space Taxonomy
Spatial Data Types and Operations
Spatial Query Languages
Algorithms for spatial operations with cost models
Spatial index access methods (with concurrency control)
Interface to DBMS
Index Structures
Spatial Join
Cost Functions Selectivity Evaluation
Bulk Loading Concurrency Control Recovery/Backup
Views Derived Data
DBMS
Object-Relational Database Servers

# GIS vs SDBMS

GIS is a specific application architecture built on top of a [more general purpose] SDBMS.

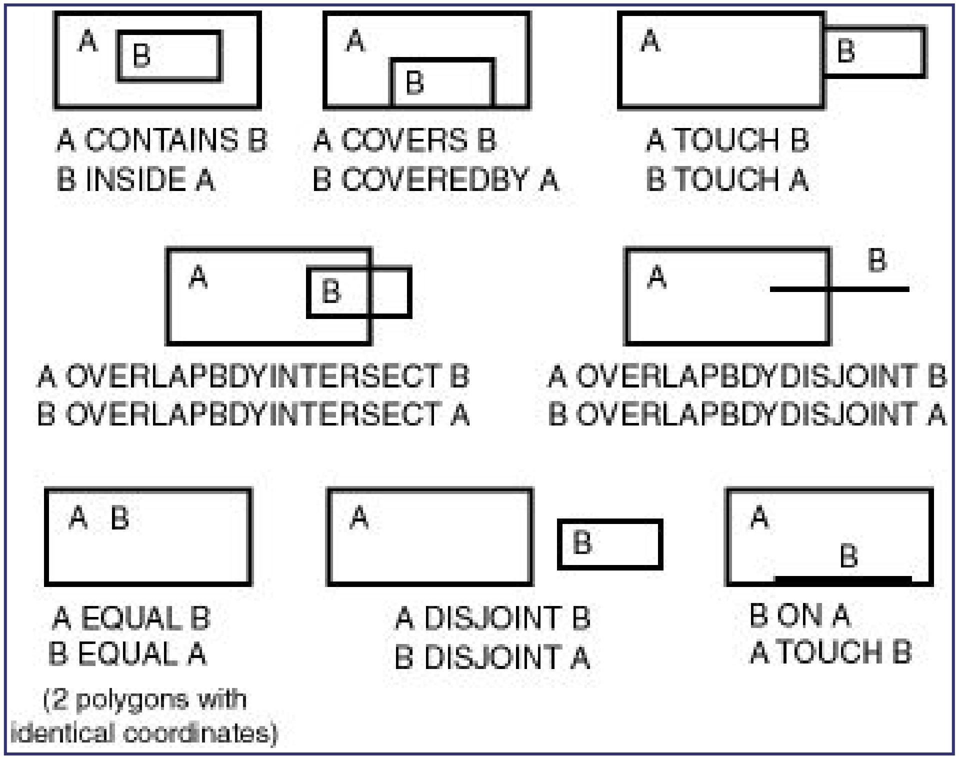
GIS typically tend to be used for:

| | |
|---|---|
| **Search** | Thematic search, search by region, (re-)classification |
| **Location analysis** | Buffer, corridor, overlay |
| **Terrain analysis** | Slope/aspect, catchment, drainage network |
| **Flow analysis** | Connectivity, shortest path |
| **Distribution** | Change detection, proximity, nearest neighbor |
| **Spatial analysis/Statistics** | Pattern, centrality, autocorrelation, indices of similarity, topology: hole description |
| **Measurements** | Distance, perimeter, shape, adjacency, direction |

# Spatial relationships

In 1D (and higher), spatial relationships can be expressed using 'intersects', 'crosses', 'within', 'touches' (these are T/F predicates).

Here is a sampling of spatial relationships in 2D:



Another diagram showing the [binary] operations:

IEEE Transactions on Knowledge and Data Engineering 6 (1): 86-95, 1994. 12

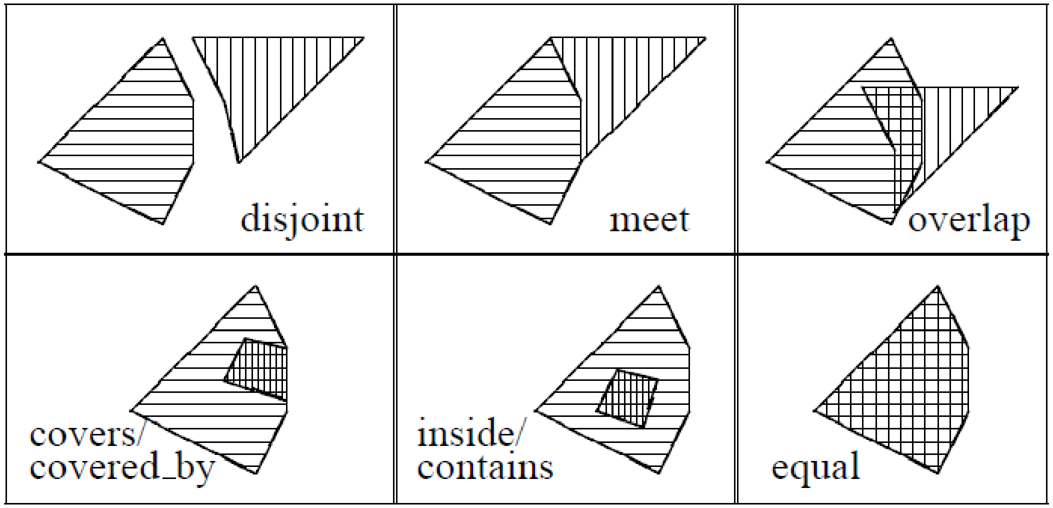
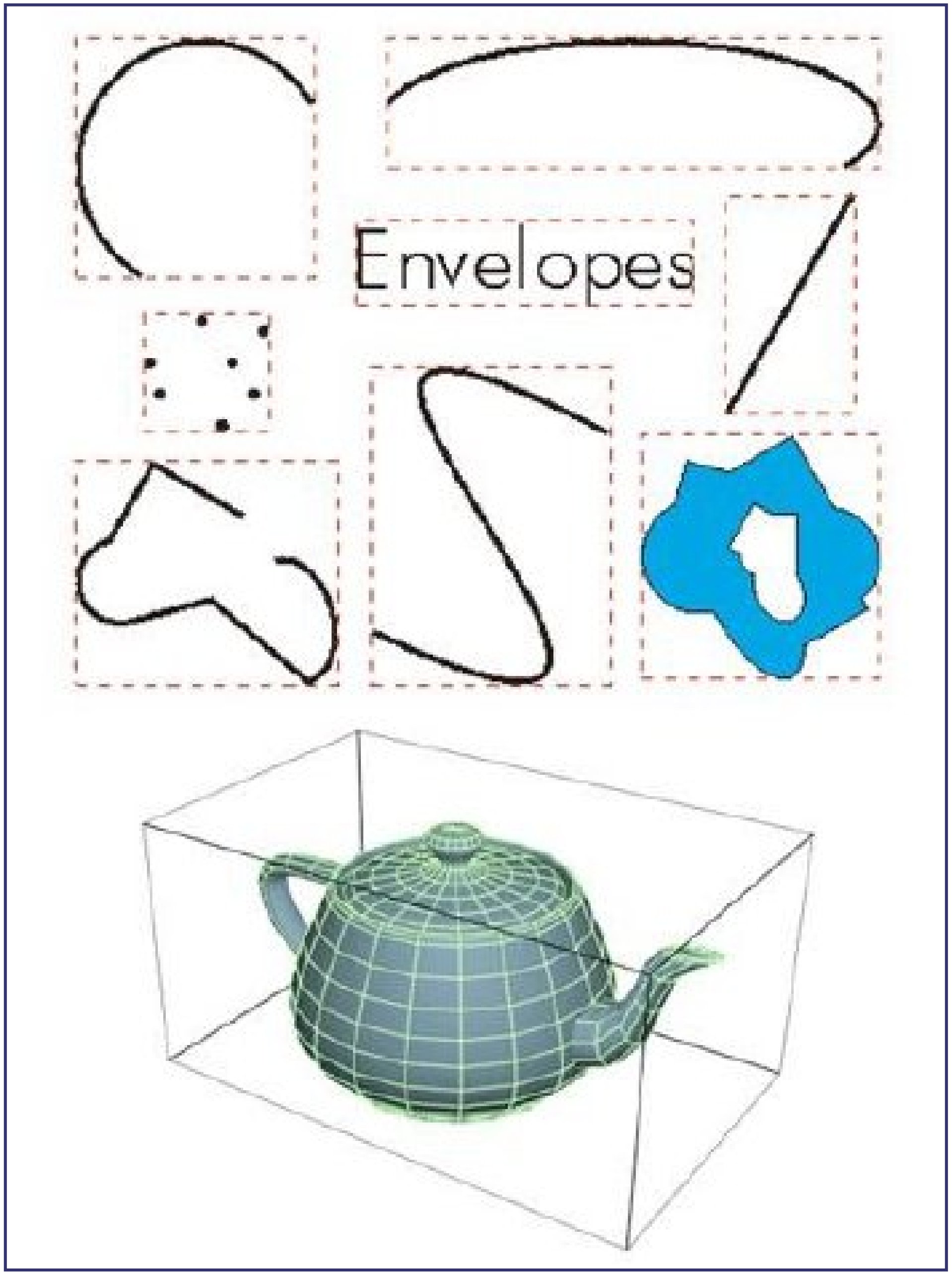



Figure 2. Examples of the binary topological relationships (a) disjoint, (b) meet, (c) overlap, (d) covers/covered_by, (e) inside/contains, and (f) equal [24].

Minimum Bounding Rectangles (MBRs) are what are used to compute the results of operations shown above:



# Spatial relations - categories

Spatial relationships can be:

- topology-based [using defns of boundary, interior, exterior]
- metric-based [distance/Euclidian, angle measures]
- direction-based
- network-based [eg. shortest path]

Topological relationships could be further grouped like so:

- proximity
- overlap
- containment

# How can we put these relations to use?

We can perform the following, on spatial data:

- spatial measurements: find the distance between points, find polygon area..
- spatial functions: find nearest neighbors..
- spatial predicates: test for proximity, containment..

## Spatial Data Entity Creation

- Form an entity to hold county names, states, populations, and geographies

```
CREATE TABLE County(
   Name      varchar(30),
   State     varchar(30),
   Pop       Integer,
   Shape     Polygon);
```

# Spatial Data Entity Creation (Cont.)

- Form an entity to hold river names, sources, lengths, and geographies

```
CREATE TABLE River(
    Name        varchar(30),
    Source      varchar(30),
    Distance    Integer,
    Shape       LineString);
```

## Example Spatial Query

- Find all the counties that border on Contra Costa county

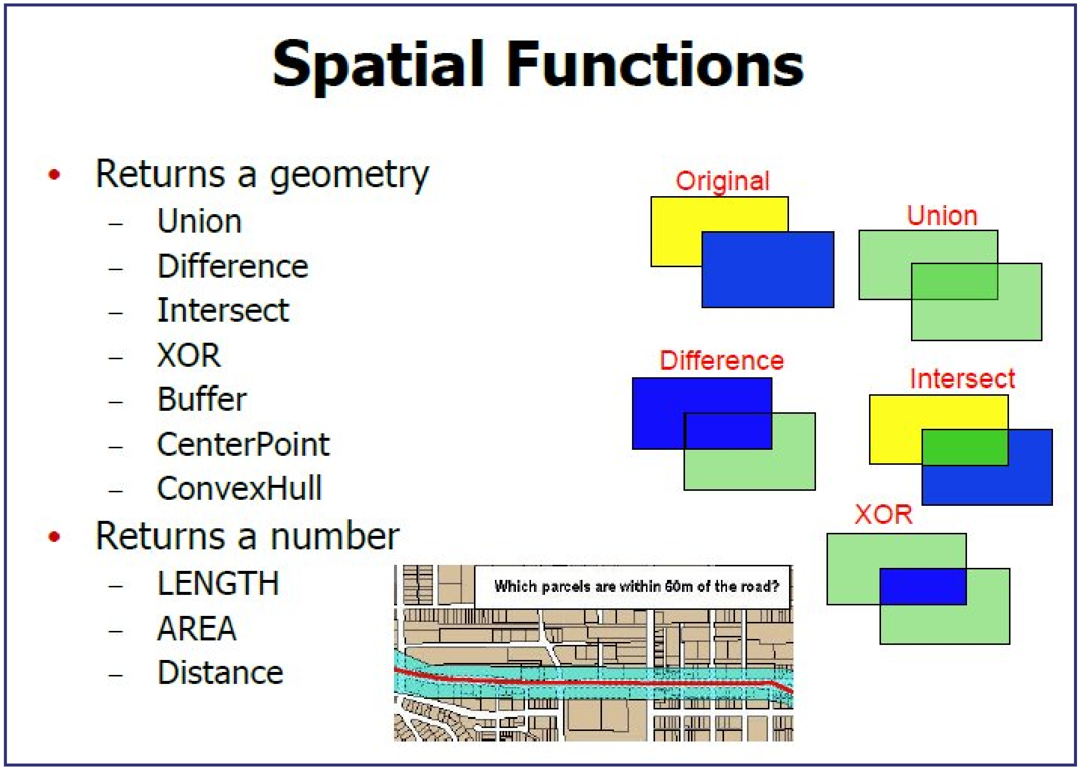
```
SELECT    C1.Name
FROM      County C1, County C2
WHERE     Touch(C1.Shape, C2.Shape) = 1
          AND C2.Name = 'Contra Costa';
```

## Example Spatial Query (Cont.)

- Find all the counties through which the Merced river runs

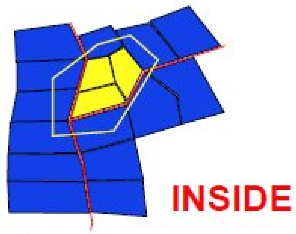
```
SELECT    C.Name, R.Name
FROM      County C, River R
WHERE     Intersect(C.Shape, R.Shape) = 1
          AND R.Name = 'Merced';
```

# Spatial operators, functions

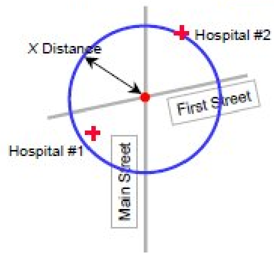


# Spatial Operators

- Full range of spatial operators
  - Implemented as functional extensions in SQL
  - Topological Operators
    - Inside Contains
    - Touch Disjoint
    - Covers Covered By
    - Equal Overlap Boundary




  - Distance Operators
    - Within Distance
    - Nearest Neighbor

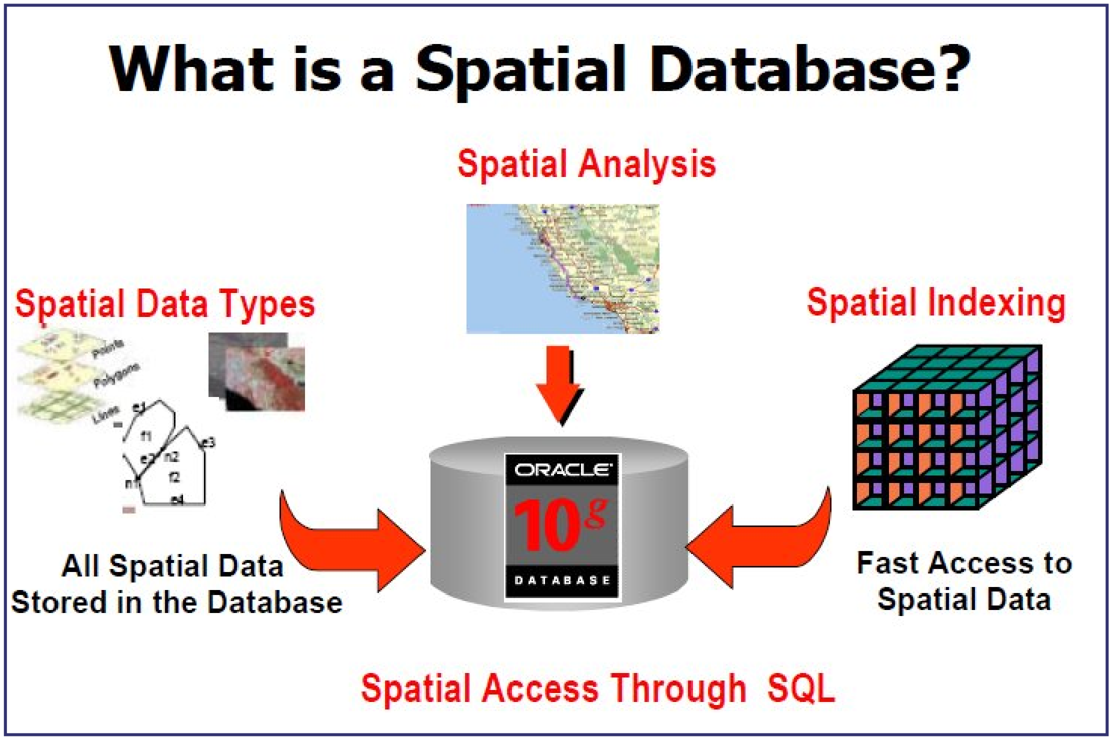


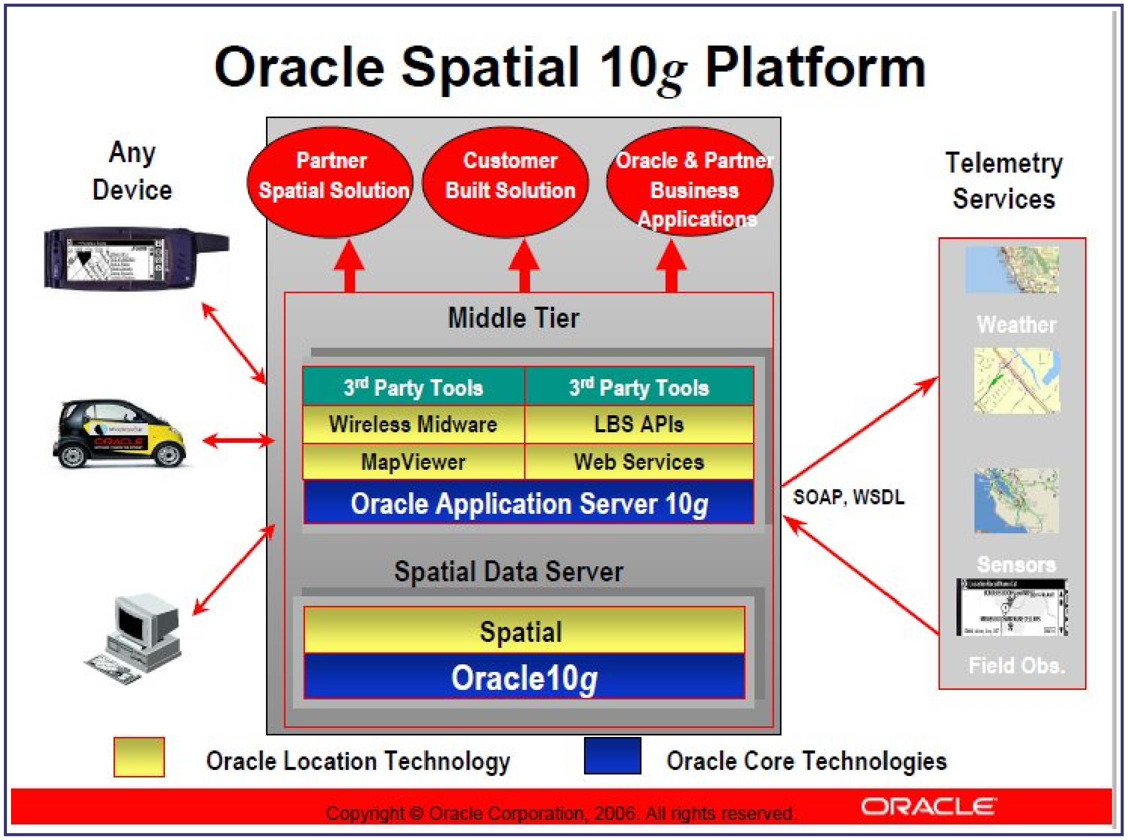
```
#query

+ equals(another :Geometry) : Boolean
+ disjoint(another :Geometry) : Boolean
+ intersects(another :Geometry) : Boolean
+ touches(another :Geometry) : Boolean
+ crosses(another :Geometry) : Boolean
+ within(another :Geometry) : Boolean
+ contains(another :Geometry) : Boolean
...

#analysis

+ distance(another : Geometry) : Distance
+ buffer(another : Distance) : Geometry
+ convexHull() : Geometry
...
```

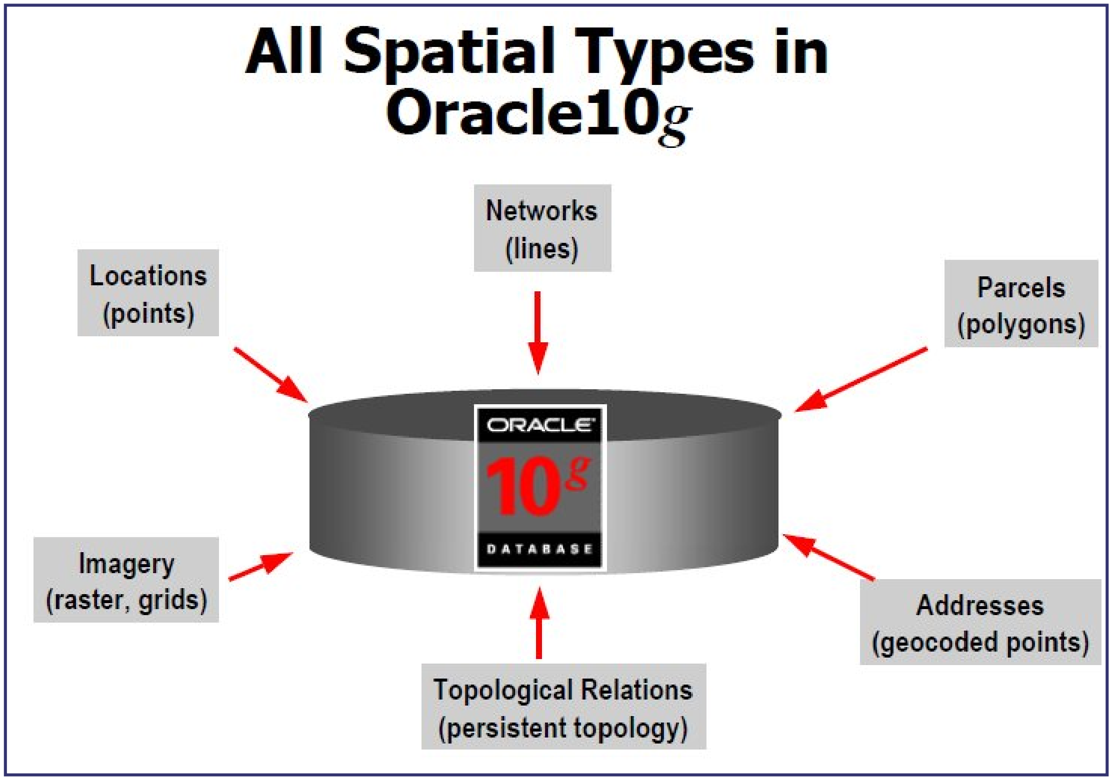
This doc [from 'FME Knowledge Center'; thanks to Minaxi Singla for the link] provides more info on the spatial operators.

# Oracle Spatial

Oracle offers a 'Spatial' library for spatial queries - this includes UDTs and custom functions to process them.




Oracle Spatial 10g Platform
Any Device
Partner Spatial Solution
Customer Built Solution
Oracle & Partner Business Applications
Telemetry Services
Middle Tier
3rd Party Tools
3rd Party Tools
Wireless Midware
LBS APIs
MapViewer
Web Services
Oracle Application Server 10g
SOAP, WSDL
Weather
Sensors
Field Obs.
Spatial Data Server
Spatial
Oracle10g
Oracle Location Technology
Oracle Core Technologies
Copyright © Oracle Corporation, 2006. All rights reserved.
ORACLE


All Spatial Types in Oracle10*g*
Networks (lines)
Locations (points)
Parcels (polygons)
ORACLE
10g
DATABASE
Imagery (raster, grids)
Addresses (geocoded points)
Topological Relations (persistent topology)

# SDO_GEOMETRY Object

- SDO_GEOMETRY Object

```
SDO_GTYPE          NUMBER
SDO_SRID           NUMBER
SDO_POINT          SDO_POINT_TYPE
SDO_ELEM_INFO      SDO_ELEM_INFO_ARRAY
SDO_ORDINATES      SDO_ORDINATE_ARRAY
```

- Example

```
SQL> CREATE TABLE states (
  2     state        VARCHAR2(30),
  3     totpop       NUMBER(9),
  4     geom         SDO_GEOMETRY);
```

# SDO_GEOMETRY Object

- SDO_GTYPE - Defines the type of geometry stored in the object

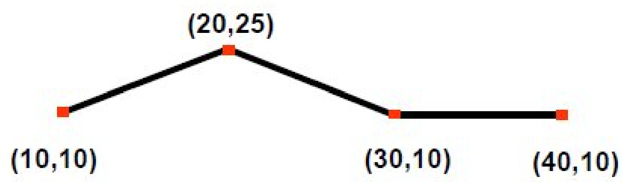
| GTYPE | Explanation |
|---|---|
| 1 POINT | Geometry contains one point |
| 2 LINESTRING | Geometry contains one line string |
| 3 POLYGON | Geometry contains one polygon |
| 4 HETEROGENEOUS COLLECTION | Geometry is a collection of elements of different types: points, lines, polygons |
| 5 MULTIPOINT | Geometry has multiple points |
| 6 MULTILINESTRING | Geometry has multiple line strings |
| 7 MULTIPOLYGON | Geometry has multiple polygons |

# SDO_GTYPE

| SDO_GTYPE | Four digit GTYPEs - Include dimensionality | | |
|---|---|---|---|
| | 2D | 3D | 4D |
| 1 POINT | 2001 | 3001 | 4001 |
| 2 LINESTRING | 2002 | 3002 | 4002 |
| 3 POLYGON | 2003 | 3003 | 4003 |
| 4 COLLECTION | 2004 | 3004 | 4004 |
| 5 MULTIPOINT | 2005 | 3005 | 4005 |
| 6 MULTILINESTRING | 2006 | 3006 | 4006 |
| 7 MULTIPOLYGON | 2007 | 3007 | 4007 |

# Constructing Geometries

```
SQL> INSERT INTO LINES VALUES (
  2>      attribute_1, …. attribute_n,
  3>      SDO_GEOMETRY (
  4>        2002, null, null,
  5>        SDO_ELEM_INFO_ARRAY (1,2,1),
  6>        SDO_ORDINATE_ARRAY (
  7>          10,10, 20,25, 30,10, 40,10))
  8>      );
```



# Spatial Operators

- Operators
  - `SDO_FILTER`
    - Performs a primary filter only
  - `SDO_RELATE` and `SDO_<relationship>`
    - Performs a primary and secondary filter
  - `SDO_WITHIN_DISTANCE`
    - Generates a buffer around a geometry and performs a primary and optionally a secondary filter
  - `SDO_NN`
    - Returns nearest neighbors

# SDO_FILTER Example

- Find all the cities in a selected rectangular area
- Result is approximate

```
SELECT c.city, c.pop90
FROM proj_cities c
WHERE sdo_filter (
        c.location,
        sdo_geometry (2003, 32775, null,
          sdo_elem_info_array (1,1003,3),
          sdo_ordinate_array (1720300,1805461,
                              1831559, 2207250))
        ) = 'TRUE';
```

**Hint 1: All Spatial operators return `TRUE` or `FALSE`. When writing spatial queries always test with = `'TRUE'`, never <> `'FALSE'` or = `'true'`.**

# SDO_RELATE Example

- Find all counties in the state of New Hampshire

```
SELECT c.county, c.state_abrv
FROM geod_counties c,
     geod_states s
WHERE s.state = 'New Hampshire'
 AND sdo_relate (c.geom,
                 s.geom,
                 'mask=INSIDE+COVEREDBY')
                    ='TRUE';
```

**Note: For optimal performance, don't forget to index** `GEOD_STATES(state)`

# Relationship Operators Example

- Find all the counties around Passaic county in New Jersey:

```
SELECT /*+ ordered */ a.county
FROM geod_counties b,
     geod_counties a
WHERE b.county = 'Passaic'
    AND b.state = 'New Jersey'
    AND SDO_TOUCH(a.geom,b.geom) = 'TRUE';
```

- Previously:

```
. . .
AND SDO_RELATE(a.geom,b.geom,
     'MASK=TOUCH') = 'TRUE';
```

# SDO_NN Example

- Find the five cities nearest to Interstate I170, ordered by distance

```
SELECT /*+ ordered */
       c.city, c.state_abrv,
       sdo_nn_distance (1) distance_in_miles
FROM geod_interstates i,
     geod_cities c
WHERE i.highway = 'I170'
  AND sdo_nn(c.location, i.geom,
         'sdo_num_res=5 unit=mile', 1) = 'TRUE'
ORDER by distance_in_miles;
```

- Note: Make sure you have an index on `GEOD_INTERSTATES (HIGHWAY)`.

# SDO_WITHIN_DISTANCE Examples

- Find all cities within a distance from an interstate

```
SELECT /*+ ordered */ c.city
FROM geod_interstates i, geod_cities c
WHERE i.highway = 'I170'
   AND sdo_within_distance (
        c.location, i.geom,
       'distance=15 unit=mile') = 'TRUE';
```

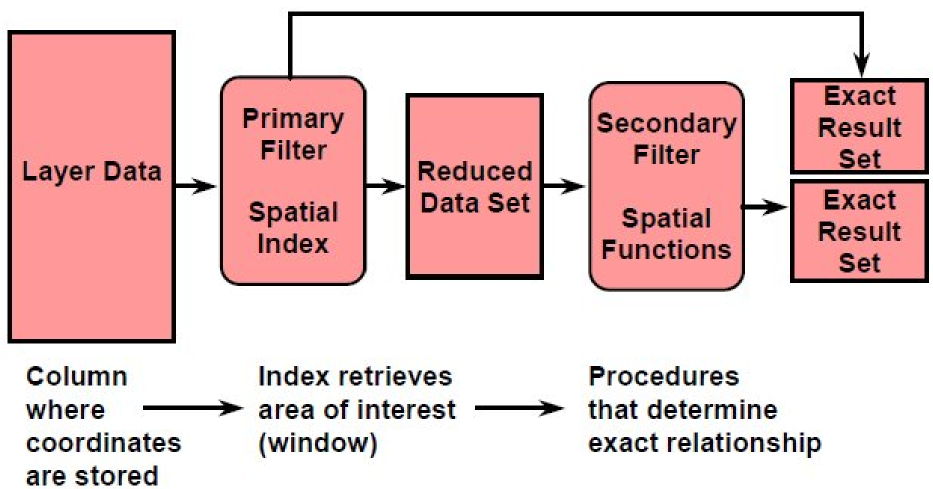
- Find interstates within a distance from a city

```
SELECT /*+ ordered */ i.highway
FROM geod_cities c, geod_interstates i
WHERE c.city = 'Tampa'
   AND sdo_within_distance (
        i.geom, c.location,
        'distance=15 unit=mile') = 'TRUE';
```

# Spatial Indexing

- Used to optimize spatial query performance
- R-tree Indexing
  - Based on minimum bounding rectangles (MBRs) for 2D data or minimum bounding volumes (MBVs) for 3D data
  - Indexes two, three, or four dimensions
- Provides an exclusive and exhaustive coverage of spatial objects
- Indexes all elements within a geometry including points, lines, and polygons

# Optimized Query Model



# Postgres PostGIS

## Types of queries - PostGIS

The function names for queries differ across geodatabases. The following list contains commonly used functions built into PostGIS, a free geodatabase which is a PostgreSQL extension (the term 'geometry' refers to a point, line, box or other two or three dimensional shape):

## Types of queries - PostGIS (Cont.)

1. Distance(geometry, geometry) : number
2. Equals(geometry, geometry) : boolean
3. Disjoint(geometry, geometry) : boolean
4. Intersects(geometry, geometry) : boolean
5. Touches(geometry, geometry) : boolean
6. Crosses(geometry, geometry) : boolean

## Types of queries - PostGIS (Cont.)

7. Overlaps(geometry, geometry) : boolean
8. Contains(geometry, geometry) : boolean
9. Intersects(geometry, geometry) : boolean
10. Length(geometry) : number
11. Area(geometry) : number
12. Centroid(geometry) : geometry

Here is an example - table creation, and polygon insertion:

```
Create Table County (
name VARCHAR(30),
shape geometry);
CREATE TABLE


Insert into County values ('Lynn', ST_Polygon(ST_GeomFromText('LINESTRING(75.15 29.53 1,77 29 1,77.6 29.5 1,
 75.15 29.53 1)'),4326));
INSERT 0 1

SELECT *
FROM County;
 name |
         shape

------+---------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
------------------
 Lynn | 01030000A0E610000001000000040000009A99999999C9524048E17A14AE873D40000000000000F03F000000000040534000
00000000003D40000000000000F03F66666666666653400000000000803D40000000000000F03F9A99999999C9524048E17A14AE873D
40000000000000F03F
(1 row)


Saty@Satys_USC_PC ~
$
```

To do the above, here are the steps on a PC (similar steps on a Mac):

- install Postgres (v.9.5, not 9.6 beta!)
- bring up 'Application Stack Builder' (an add-on that gets installed when Postgres v9.5 is installed), from the available installation options that come up, pick Spatial Extensions -> 'PostGIS 2.2 for Postgres 9.5', install
- bring up a shell (I use 'cygwin'); note - if you want to use cygwin, be sure to use the shell that comes up when you run cygwin.bat, *not* the 'mintty' shell that you get when you double-click on the cygwin icon; Mac users would use the built-in shell
- 9.5/bin/initdb (on a Mac the path would be different)
- 9.5/bin/pg_ctl start - this starts the Postgres server
- 9.5/bin/createdb mydb - a new db for us to create tables in
- 9.5/bin/psql.exe -d mydb -c "CREATE EXTENSION postgis;" - this adds spatial types to our db; note: 'psql' is the program that lets us communicate with the db server, via the shell
- 9.5/bin/psql.exe -d mydb -a -f county.sql - this is how you can execute SQL commands that you store in a .sql file

- edit the .sql file (eg add more data [including spatial data], create new tables, write SQL queries [including spatial ones]..), run the file (as shown above), edit, run......
- 9.5/bin/pg_ctl stop - optionally you can stop the server and restart it later
- ...

You can learn a lot about spatial queries from this page.

# Creating spatial indexes

As (more so than) with non-spatial data, the creation and use of spatial indexes VASTLY speed up processing!

# Can B Trees index spatial data?

In short, YES, if we pair it up with a 'z curve' indexing scheme (using a space-filling curve):

## Organizing spatial data with space filling curves

- Issue:
  - Sorting is not naturally defined on spatial data
  - Many efficient search methods are based on sorting datasets
- Space filling curves
  - Impose an ordering on the locations in a multi-dimensional space
    - Examples: row-order (Fig. 1.11(a), z-order (Fig 1.11(b))
  - Allow use of traditional efficient search methods on spatial data

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

(a)

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

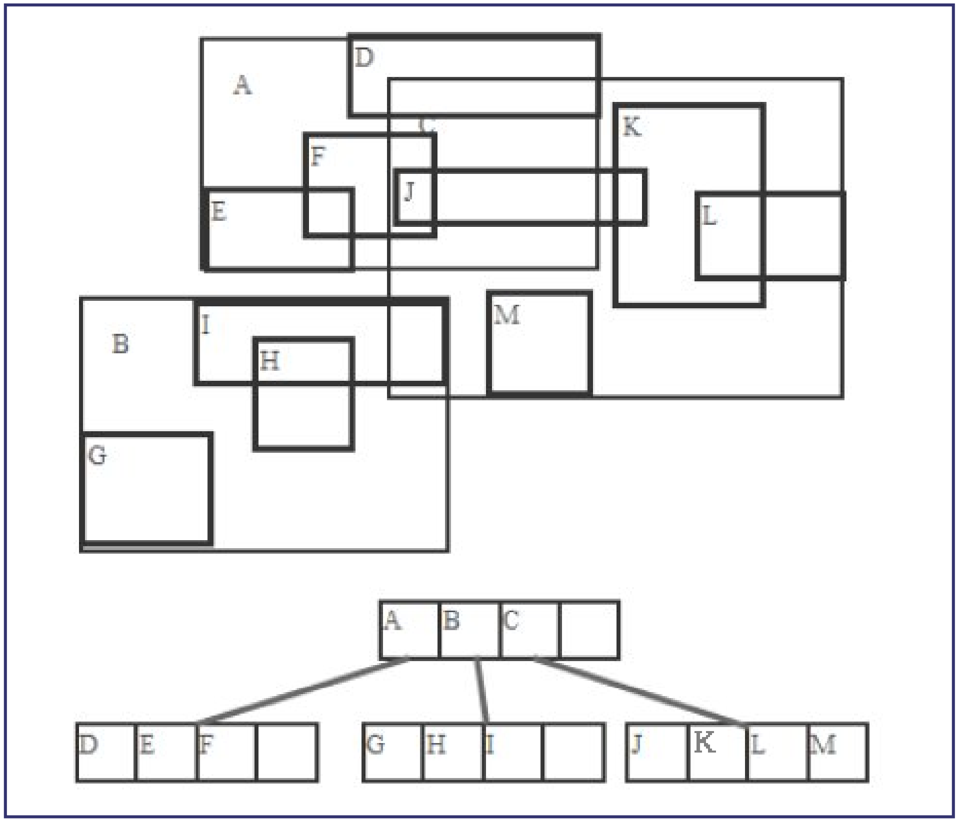
(b)

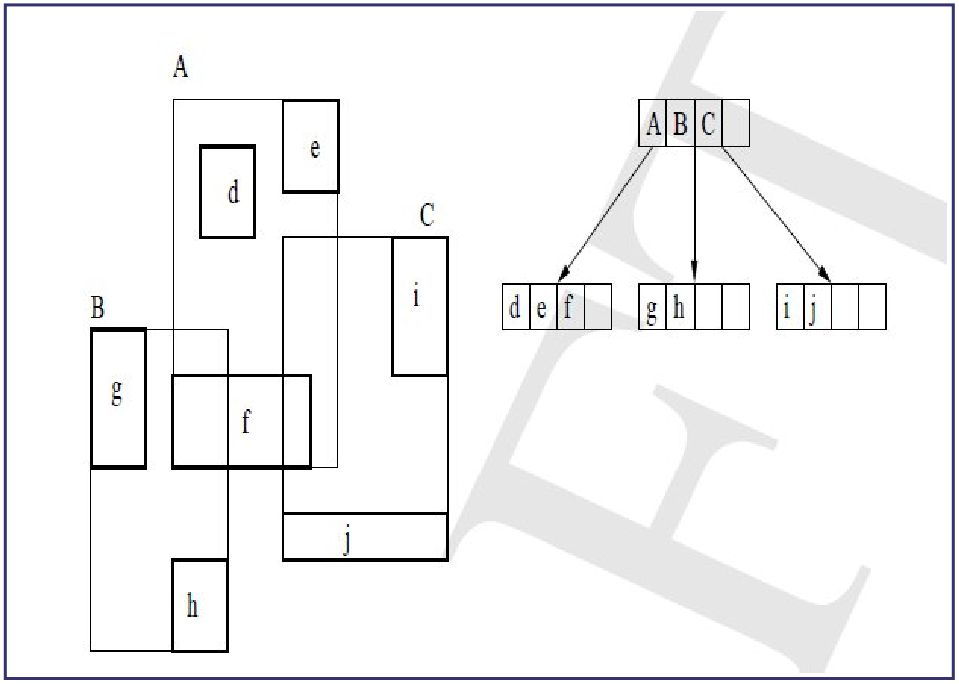
0:00 / 0:30

The idea is to quantize every (x,y) location into a recursively-divided 'quadtree' cell, and use the cell's binary (x,y) location to create a (binary) 'z' key, which is ordered along the unit (0..1) interval - in other words, 2D (x,y) points get mapped (indexed) to ordered 1D 'z' locations.

But, this is of academic interest mostly, not commonly practiced in industry - Apple's FoundationDB is an exception.

# R trees



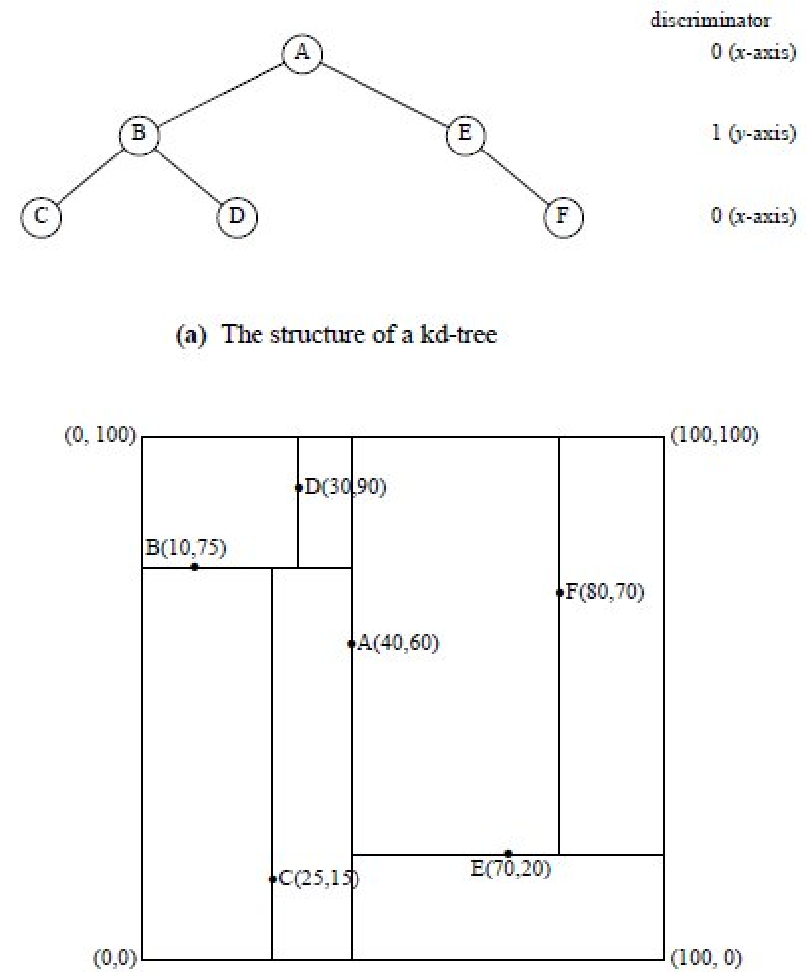



R trees use MBRs to create a hierarchy of bounds.

Variations, FYI: R+ tree, R* tree, Buddy trees, Packed R trees..

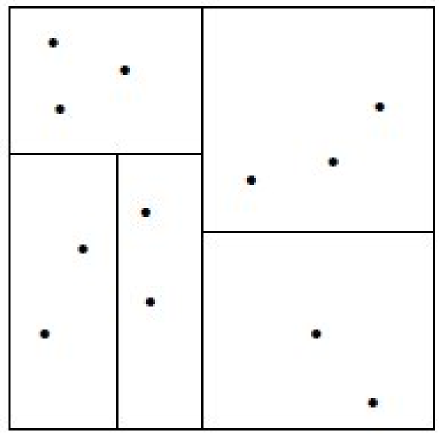
# k-d trees, K-D-B trees

k-d tree

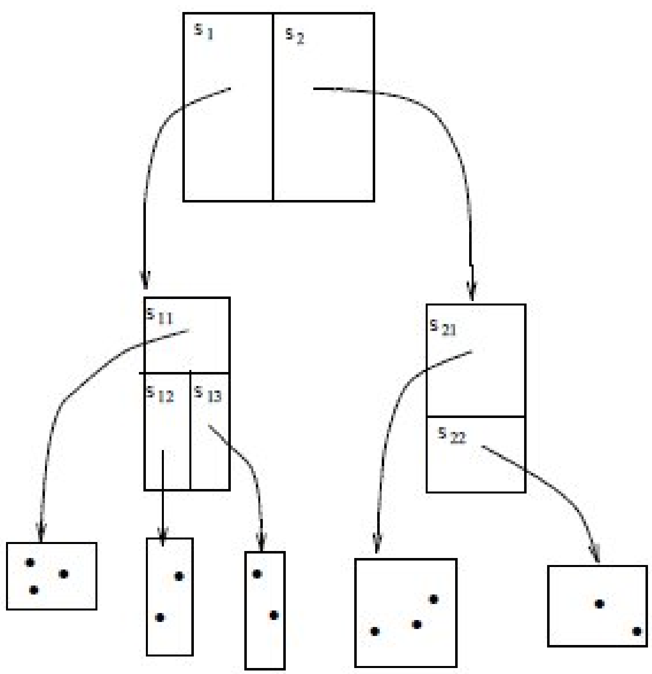



(a) The structure of a kd-tree

Alternate: K-D-B tree:



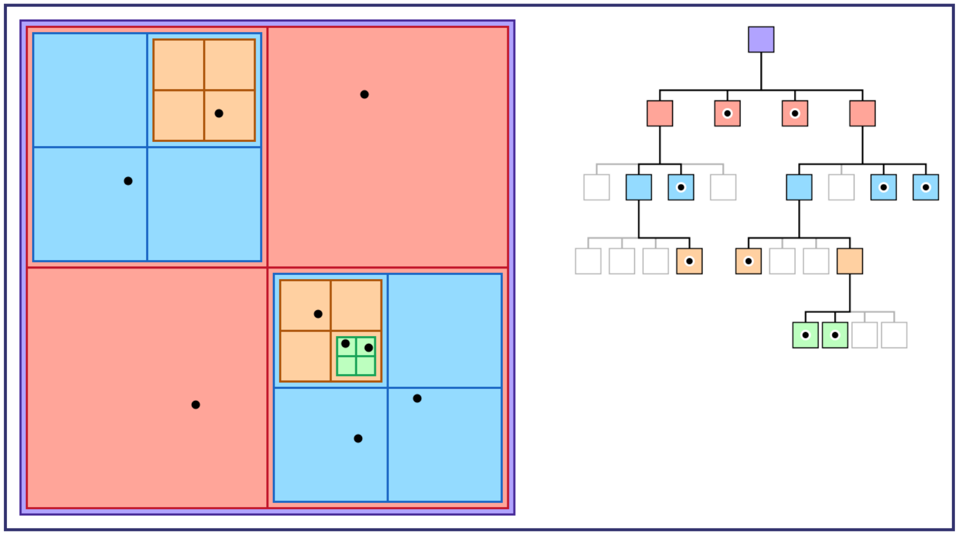
(a) Planar partition

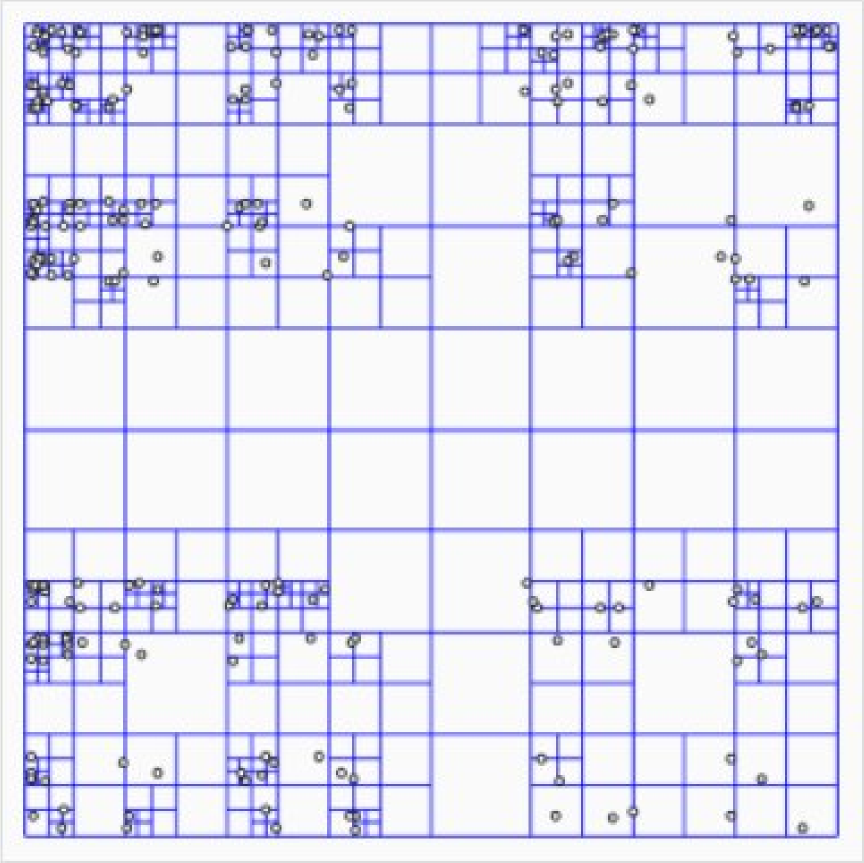



(b) A hierarchical K-D-B-tree structure

# Quadtrees (and octrees)

Here we recursively and adaptively subdivide space [subdivisions happen only where necessary].

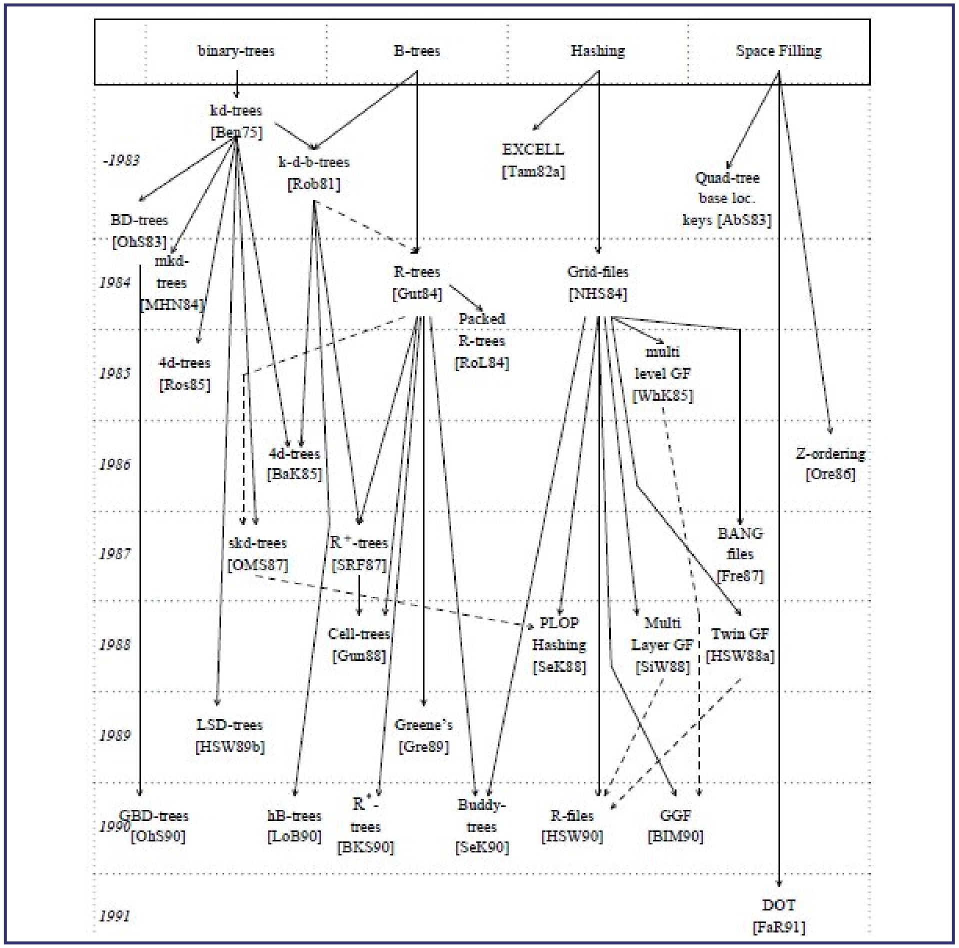




A representation of how a quadtree divides an indexed area. Source: Wikipedia

Each node is either a leaf node, with indexed points or null, or an internal (non-leaf) node that has exactly 4 children. The hierarchy of such nodes forms the quadtree.

# Indexing evolution




Indexing schemes continue to evolve.

# Query processing: filter, refine

## Query Processing

- Efficient algorithms to answer spatial queries
- Common Strategy - filter and refine
  - Filter Step:Query Region overlaps with MBRs of B,C and D
  - Refine Step: Query Region overlaps with B and C

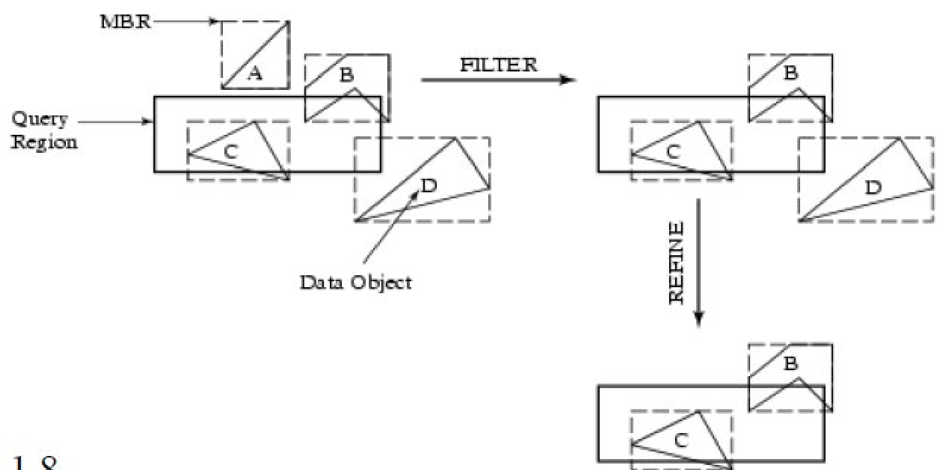
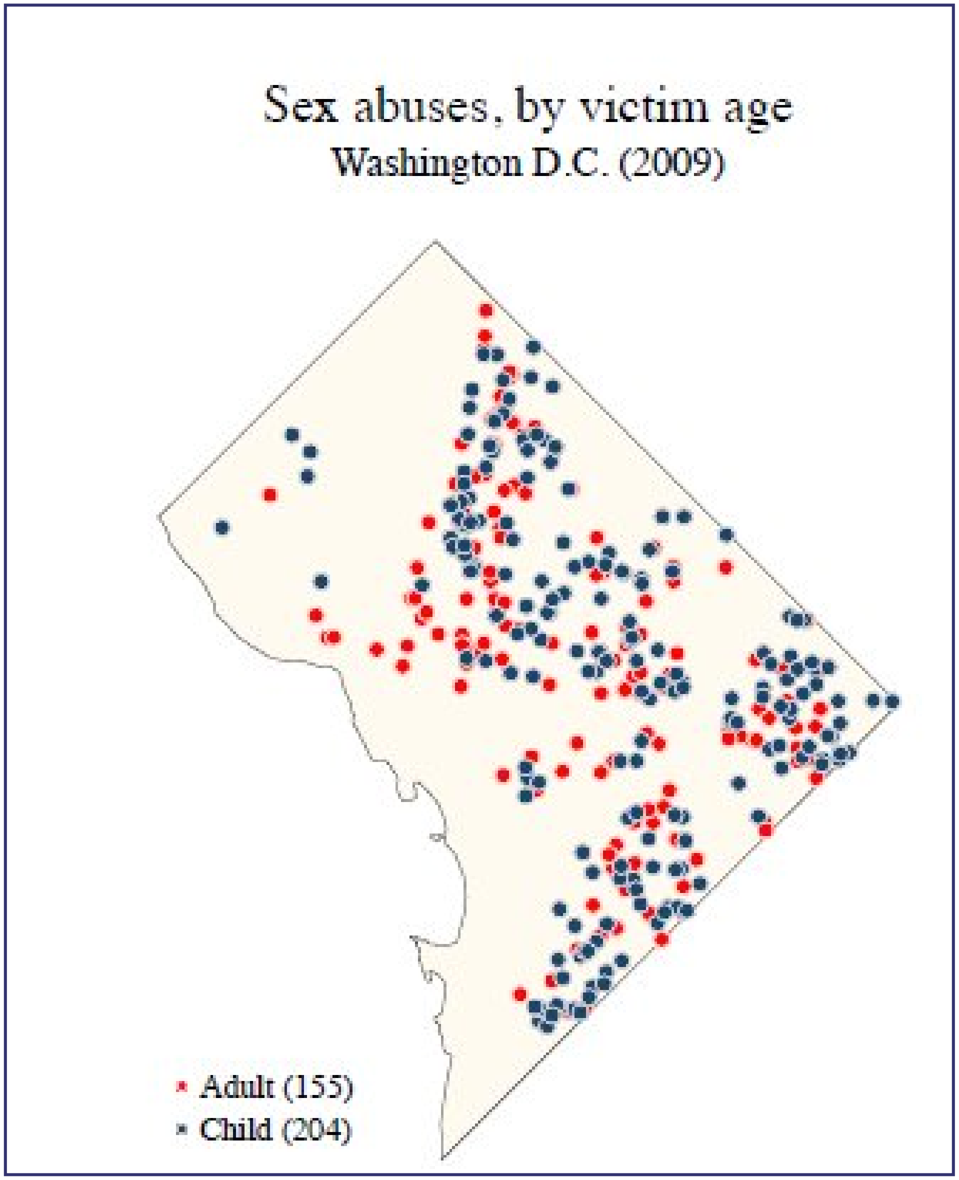



Fig 1.8

# Visualizing spatial data

A variety of non-spatial attrs can be mapped on to spatial data, providing an intuitive grasp of patterns, trends and abnormalities. Following are some examples.

Dot map:



Here's another one.

Proportional symbol map:

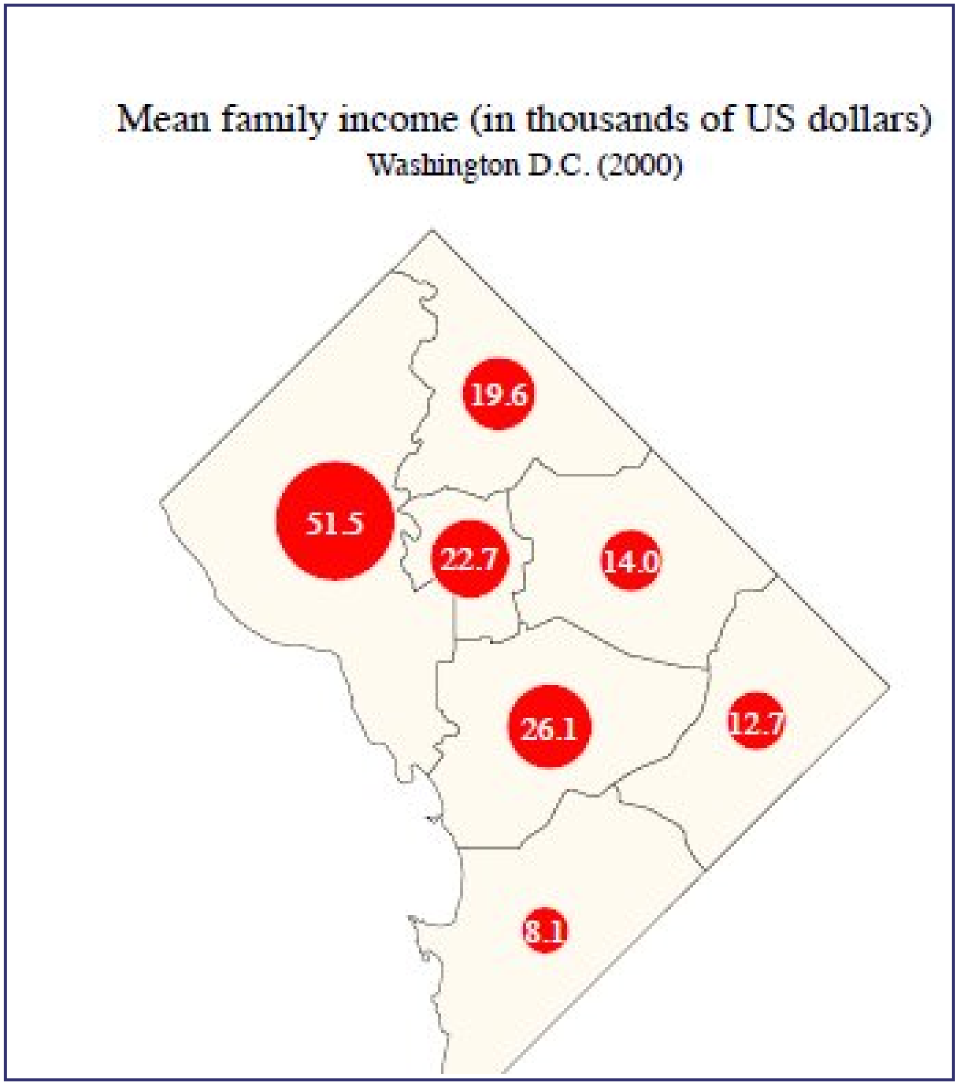
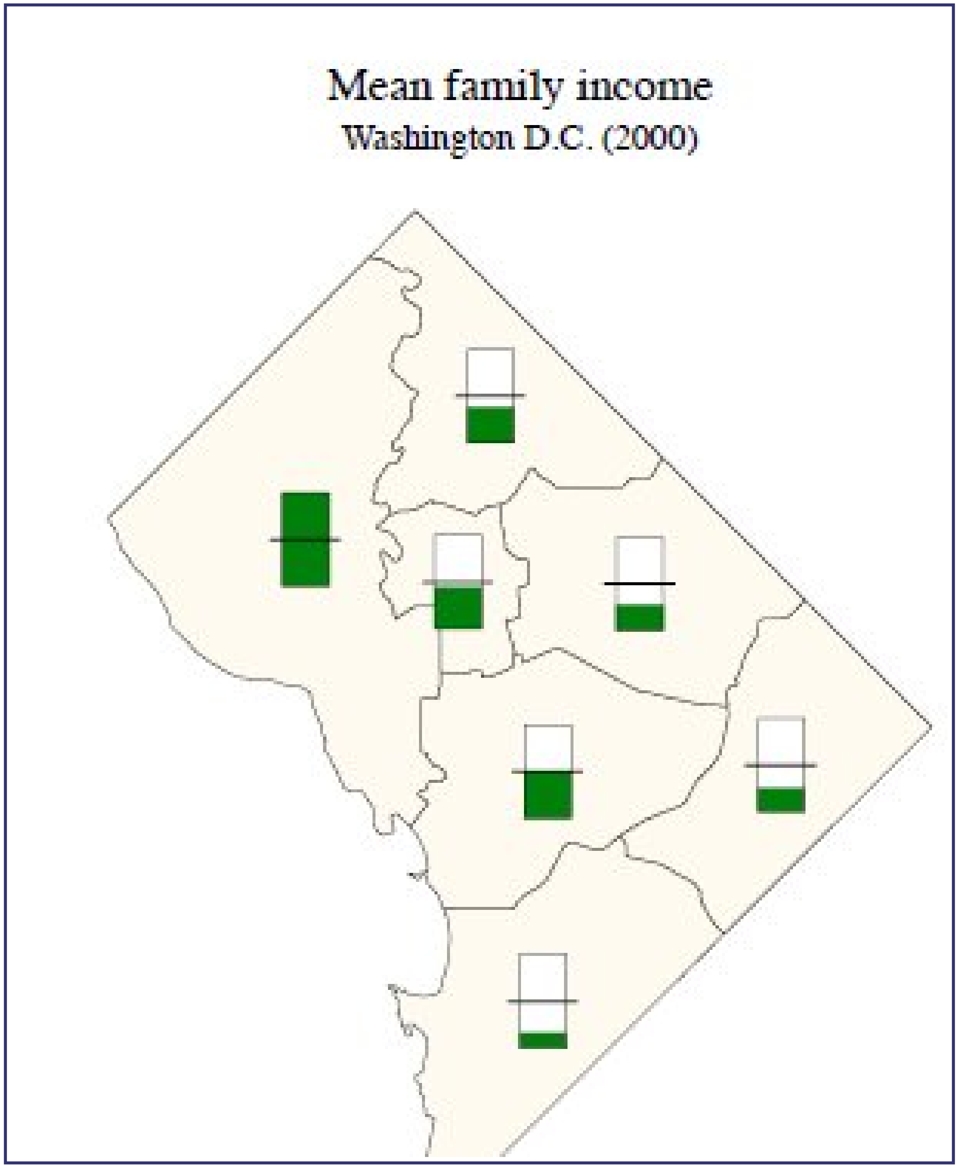
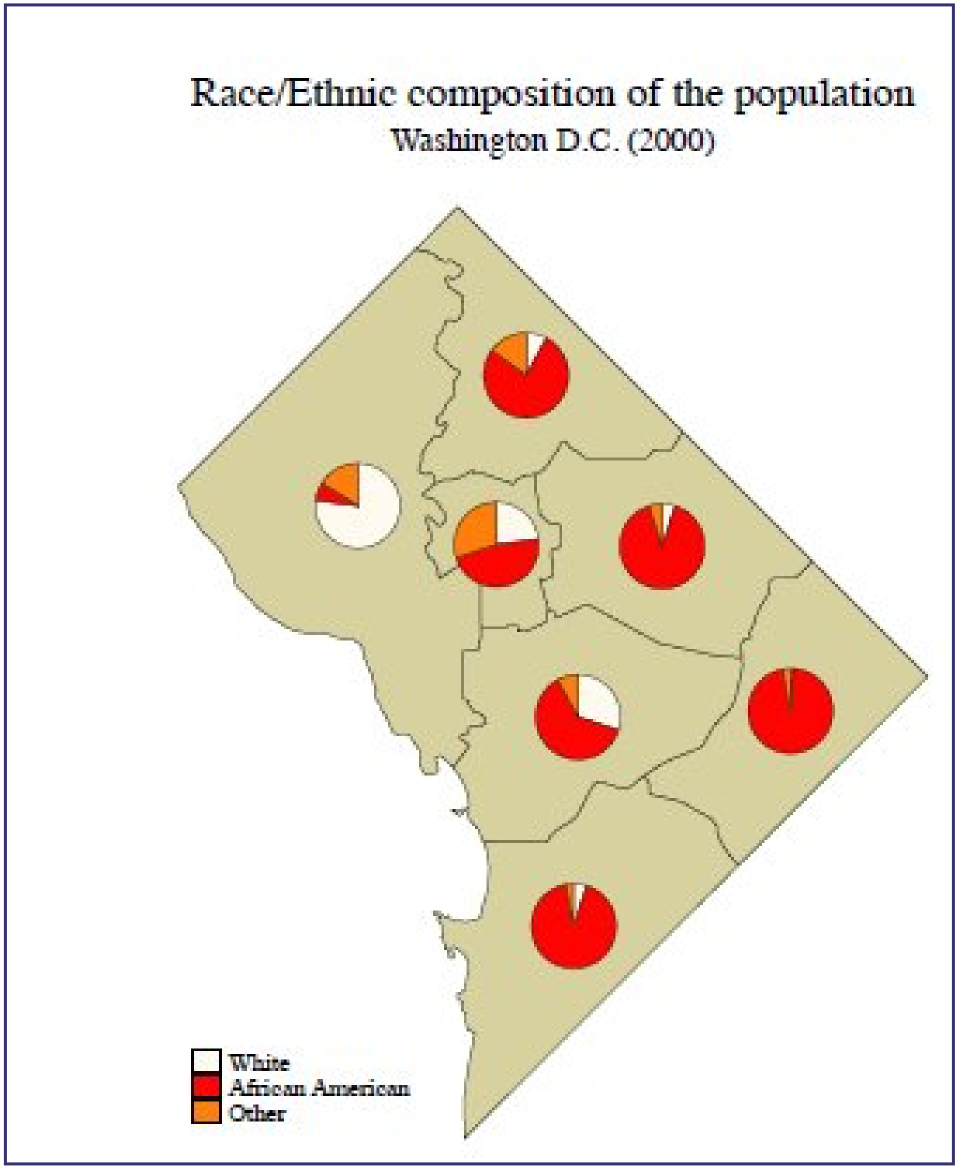


Diagram map:

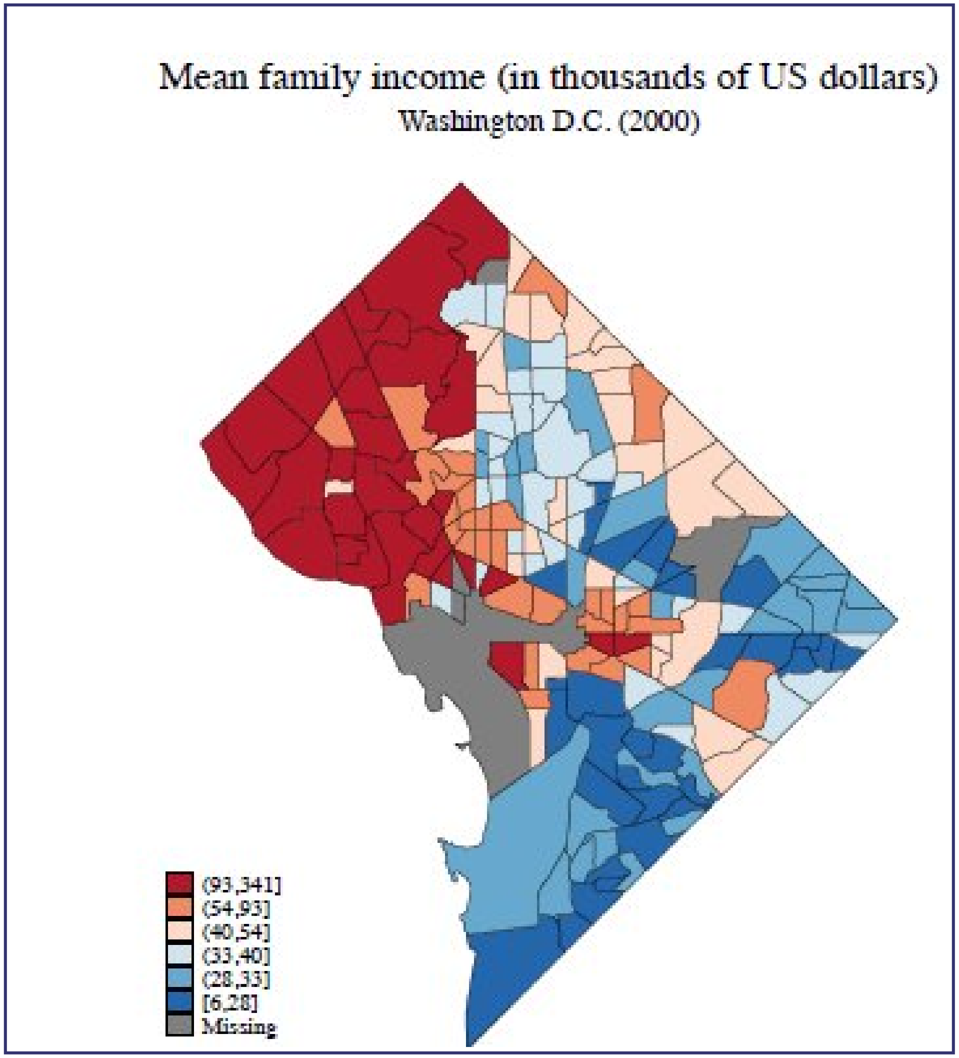


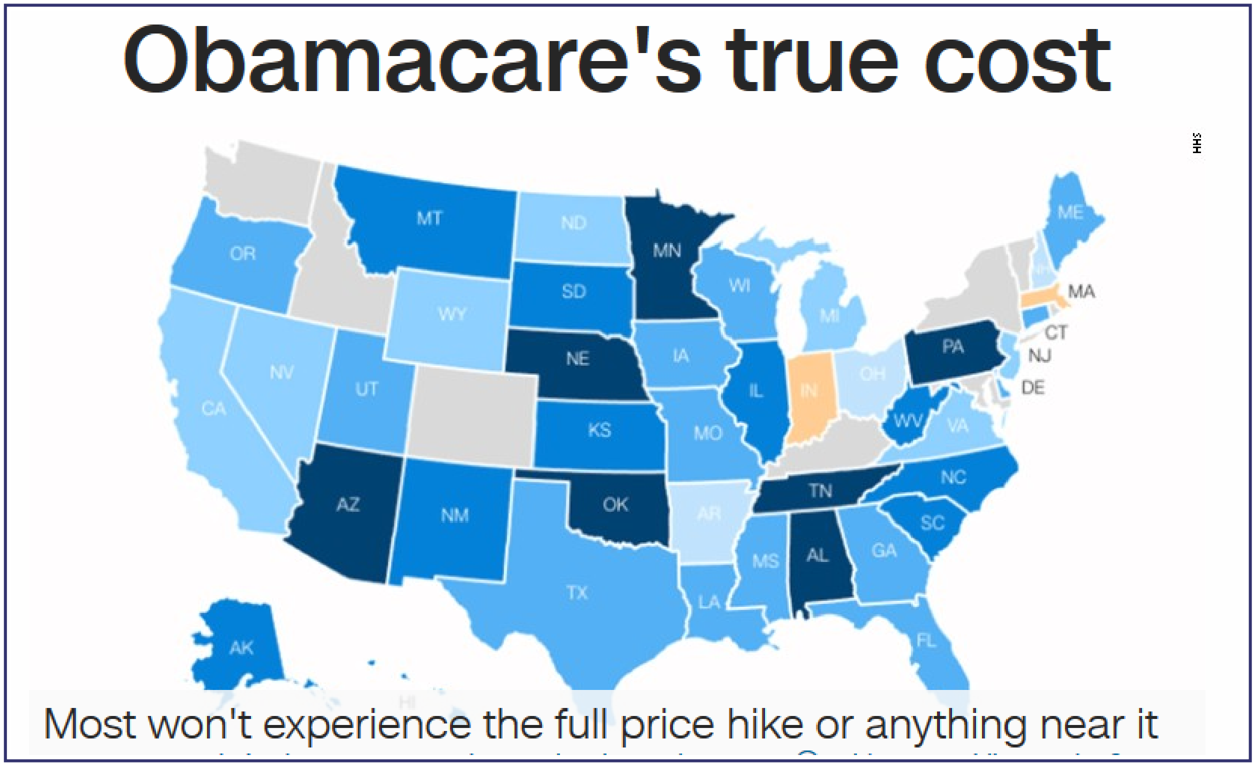
Another diagram map:




Also possible to plot multivariate data this way.

Choropleth maps (plotting of a variable of interest, to cover an entire region of a map):




Obamacare's true cost
MT
ND
MN
ME
OR
SD
WI
MI
MA
WY
NE
IA
PA
CT
NJ
NV
UT
IL
IN
OH
DE
CA
KS
MO
WV
VA
NC
TN
AZ
OK
AR
NM
SC
AL
GA
MS
TX
LA
FL
AK
Most won't experience the full price hike or anything near it

# So who (else) has spatial extensions?

Everyone!

Thanks to SQL's facility for custom datatype ('UDT') and function creation ('functional extension'), "spatial" has been implemented for every major DB out there:

- Oracle: Locator, Spatial, SDO
- Postgres: PostGIS
- DB2: Spatial Datablade
- Informix: Geodetic Datablade
- SQL Server: Geometric and Geodetic Geography types
- MySQL: spatial library comes 'built in'
- SQLite: SpatiaLite
- ..

# Google KML

Google's KML format is used to encode spatial data for Google Earth, etc. Here is a page on importing other geospatial dataset formats into Google Earth.

# OpenLayers

OpenLayers is an open GIS platform.

# ESRI: Arc*

ESRI is the home of the powerful, flexible family of ArcGIS products - and they are local!

# QGIS etc.

There is a variety of inexpensive/open source mapping platforms, competing with more pricey commercial offerings (from ESRI etc). Here are several:

- QGIS
- MapBox
- Carto
- GIS Cloud