

# Chapter 5: What is Where?

- 5.1 Basic Database Management
- 5.2 Searches By Attribute
- 5.3 Searches By Geography
- 5.4 The Query Interface

# Definition 1: A GIS is a toolbox

*"a powerful **set of tools** for storing and retrieving at will, transforming and displaying spatial data from the real world for a particular set of purposes" (Burrough, 1986, p. 6).*

*"automated **systems** for the capture, storage, retrieval, analysis, and display of spatial data." (Clarke, 1995, p. 13).*

# Definition 1: A GIS is a toolbox

- **Virtual Map Storage** – Maps as numbers
- **Capture** – Getting the map into the computer
- **Retrieval** – What is where

# A GIS can answer the question: What is where?

- **WHAT:** Characteristics of attributes or features.
- **WHERE:** In geographic space.

# *Flat File Database*

|        | Attribute | Attribute | Attribute |
|--------|-----------|-----------|-----------|
| Record | Value     | Value     | Value     |
| Record | Value     | Value     | Value     |
| Record | Value     | Value     | Value     |

# Arc/node map data structure with files

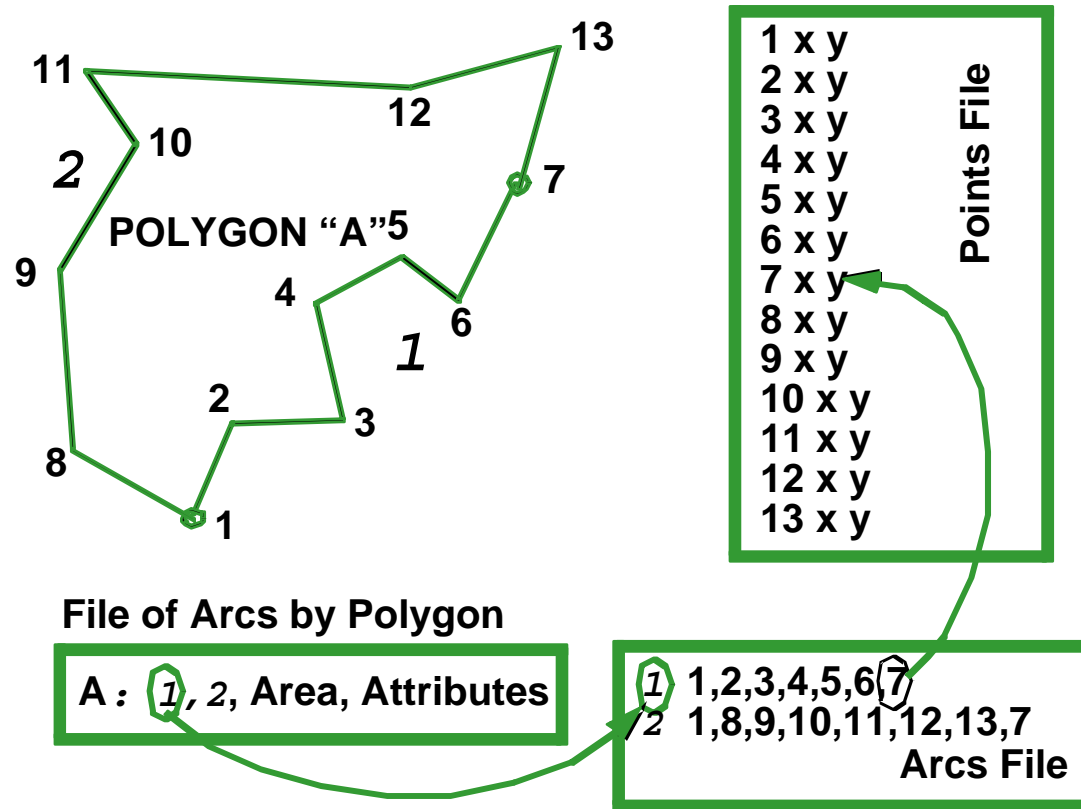


Figure 3.4 Arc/Node Map Data Structure with Files.

# A GIS links attribute and spatial data



- **Attribute Data**

- Flat File
- Relations

- **Map Data**

- Point File
- Line File
- Area File
- Topology
- Theme

# The Two Types of Data in GIS

**Spatial data:** Describing **where** things are

**AND**

**Attribute data:** Describing **what** things are

- Example: A point specified by UTM coordinates
  - Easting = 50,000 m
  - Northing = 5,000,000 m
  - Zone = 17
- This specifies the **location** of a point of the ground
- The nature of the real-world feature located at this point would be recorded in the **attribute data**
- Traditionally, geographic data and attributes were recorded on paper too (maps), and these had the same problems as a phone book



# What is a Data Model?

- A **logical construct** for the storage and retrieval of information.
- GIS map data structures are **map data models**
  - **Raster and Vector** Data Models
- **Attribute data models** are needed for the **DBMS**.
- The origin of DBMS data models is in **computer science**.

# GIS Database Models

- The approaches used for constructing GIS database management systems have depended upon the **development of DBMS** in computer science. This dates back to the 1970s when data entry used punch cards, and it has come a long way since then ...
- The first successful GIS Arc/INFO was really the marriage of two separate components:
  - The Arc **spatial data** processing component
  - The INFO **relational database management system**

# Some Database Definitions

- **Database** – an integrated set of data on a particular subject
- **Geographic (~spatial) database** - database containing geographic data of a particular subject for a particular area
- **Database Management System (DBMS)** – software to create, maintain and access databases

# Advantages of Databases over Files

- DBs avoid **redundancy** and duplication
- DBs reduce data maintenance **costs**
- Applications are separated from the data
  - Applications **persist** over time
  - Support multiple concurrent applications
- DBs facilitate better **data sharing**
- Security and standards can be defined and enforced using DBs

# Disadvantages of Databases over Files

- **Expense** of databases
- **Complexity** of databases
- **Performance** of databases – especially with complex data types (including spatial data)
- **Integration** with other systems can be difficult, especially if those systems don't use the same data model

# Characteristics of DBMS (1)

- Data model support for **multiple data types**
  - e.g MS Access: Text, Memo, Number, Date/Time, Currency, AutoNumber, Yes/No, OLE Object, Hyperlink, Lookup Wizard
- **Load data** from files, databases and other applications
- **Indexed** for rapid retrieval

# Characteristics of DBMS (2)

- Query language – SQL provides a **structured way** to ask questions of the data
- Security – **controlled access** to data
  - Multi-level groups etc.
- Controlled **update** using a transaction manager manages the updating process
- **Backup and recovery** of data for when the unthinkable happens ...
- DBA tools for optimizing performance
  - Configuration, tuning

# A DBMS contains:

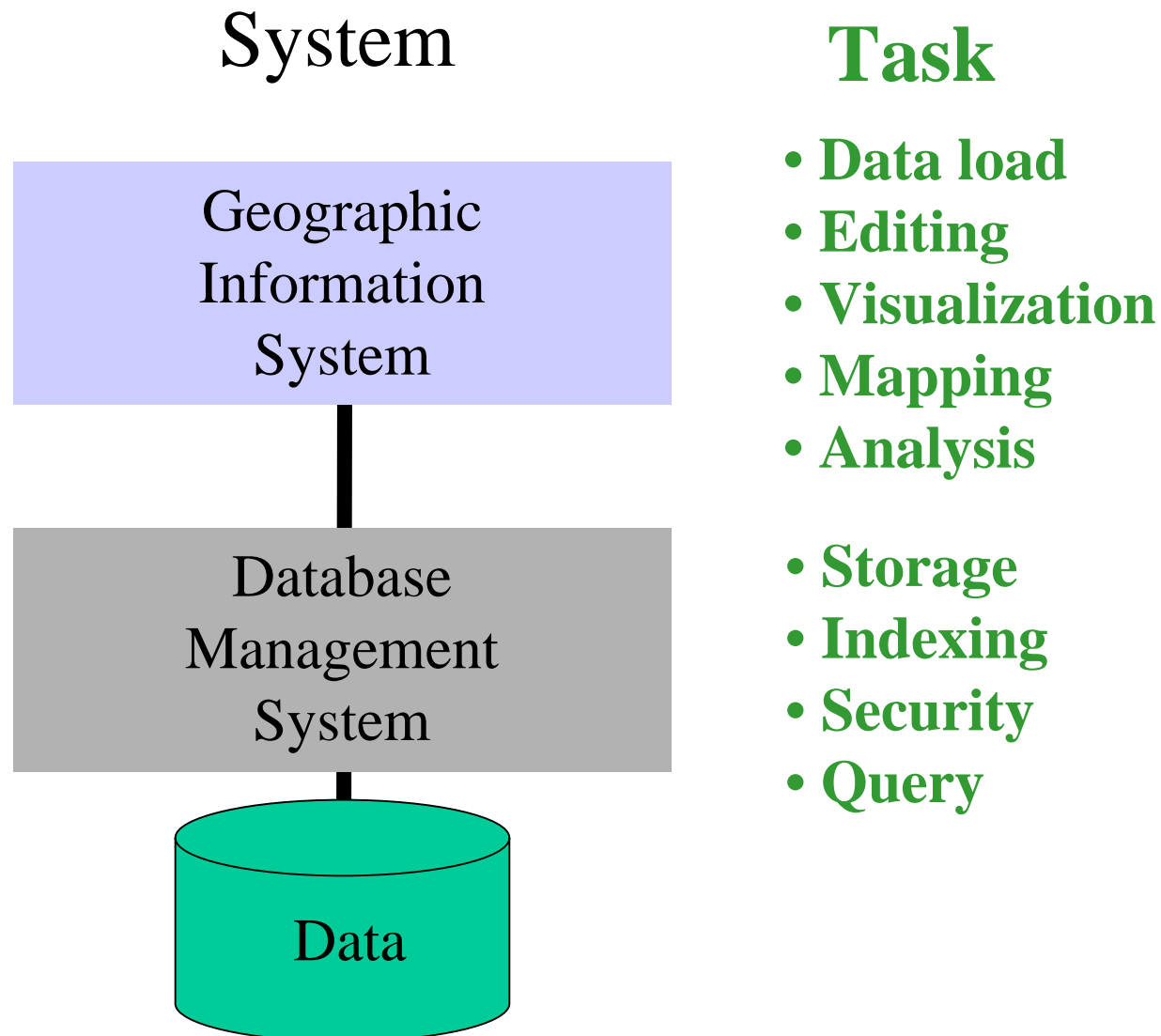
- **Data definition language**
  - Set up a database
  - Define attribute fields, their types, and lengths
  - Define user permission
- **Data dictionary**
  - A catalog of all attributes with their legal values and ranges
- **Data-entry module**
  - Most basic management function
  - Should enforce the ranges and limits defined by the DFL
- **Data update module**
  - Deletion, insertion and modification of records
- **Report generator**
- **Query language** – SQL (Structured Query Language)



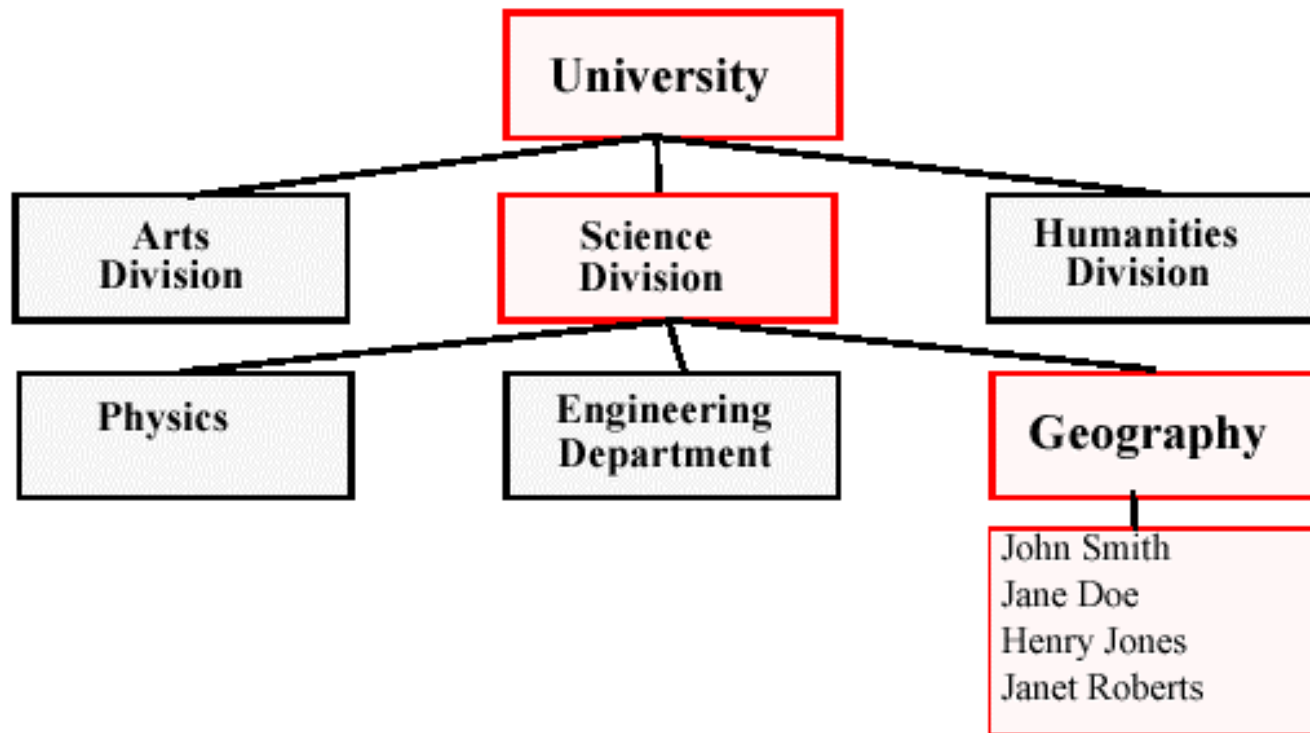
# GIS and DBMS

- Ability of the DBMS or GIS to **get back** on demand data that were previously stored.
- **Geographic search** is the secret to GIS data retrieval.
- Many forms of data organization are **incapable** of geographic search.
- Geographic information systems have **embedded** DBMSs, or link to a commercial DBMS.
- **Examples:** Access, SQL Server, ORACLE, Excel

# The Role of DBMS in GIS

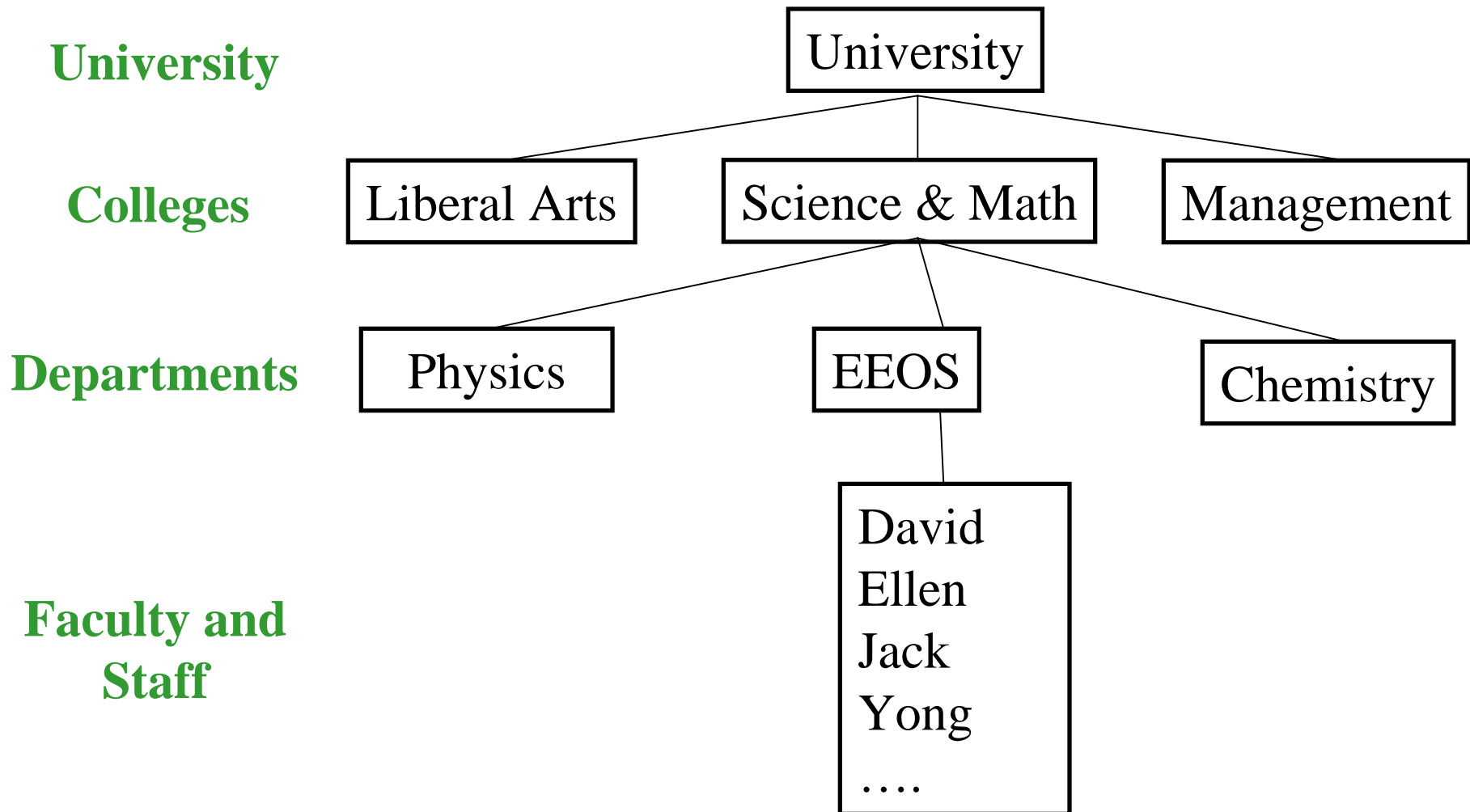


# Historically, databases were structured hierarchically in files...



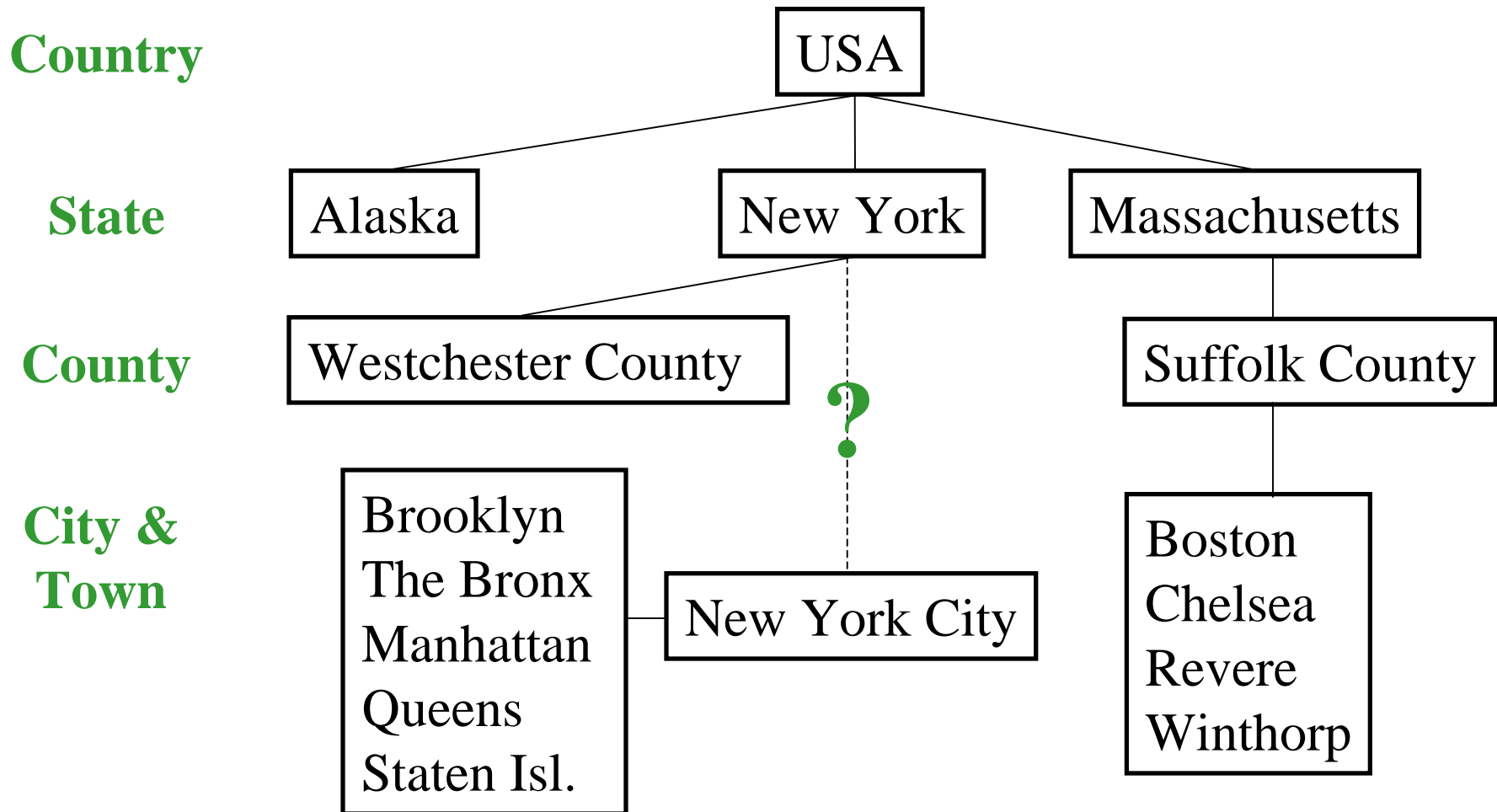
# Hierarchical Data Model

- Suppose we are designing a model for faculty & staff data:



# Hierarchical Data Model

- Now, suppose we are creating a model for places in the USA



# Disadvantages of the Hierarchical Data Model

- The database is defined in terms of a **tree structure** which is inflexible and has trouble dealing with exceptions (i.e. all records need to follow the same **uniform** structure):
  1. We **cannot define new linkages** between records once the hierarchical tree is established
  2. We **cannot define linkages laterally** or **diagonally** in the tree, only vertically
  3. The only **geographical relationships** which can be **encoded easily** are “is contained within” or “belongs to”

# Types of DBMS Models

- Hierarchical
- Network
- Relational - RDBMS ← **Our focus**
- Object-oriented - OODBMS
- Object-relational - ORDBMS

# Most current GIS DBM is by relational databases.

- Based on **multiple flat files** for records
- Connected by a **common key attribute**.
- Key is a **UNIQUE** identifier at the “atomic” level for every record (Primary Key)



# Relational Data Bases

| Patient Record |          |           |          |
|----------------|----------|-----------|----------|
| Key            | Check-in | Check Out | Room No. |
| 42             | 2/1/98   | 2/4/96    | N763     |
| 78             | 2/3/98   | 2/4/96    | N712     |

FILE

| Purchase Record |        |       |             |     |  |
|-----------------|--------|-------|-------------|-----|--|
| Item            | Date   | Price | Customer    | Key |  |
| Skate Board     | 2/1/98 | 49.95 | John Smith  | 42  |  |
| Baseball Bat    | 2/1/98 | 17.99 | James Brown | 78  |  |

FILE

| Accident Report |            |              |     |                   |
|-----------------|------------|--------------|-----|-------------------|
| Date            | Injury     | Name         | Key | Location          |
| 2/1/98          | Broken Leg | John Smith   | 42  | 75 Elm Street     |
| 2/2/98          | Concussion | Sylvia Jones | 654 | 12 State Street   |
| 2/2/98          | Cut on Ear | Robert Doe   | 123 | 2323 Broad Street |

FILE

# Relational Data Model

The **relational model** organizes data in a series of two-dimensional tables, each of which contains records for one kind of entity

Fields →

records ↓

| PID #   | Name  | Major | Phone #  | ... |
|---------|-------|-------|----------|-----|
| 1010789 | John  | EEOS  | 555-4321 | ... |
| 1021384 | David | Comm. | 555-6789 | ... |

This model is a **revolution** in database management →  
It replaced almost all other approaches in database management because it allows more **flexible relations** between kinds of entities

# Relational DBMS


- In a RDBMS, data is stored as **tuples** (pronounced tup-el), and is conceptualized as tables
- **Table** – contains data about a class of objects
  - Two-dimensional list (array)
  - Rows = objects
  - Columns = object states (properties, attributes)

# A Table

Column = property

Table =  
Object Class

Row = object



The screenshot shows a software window titled "Attributes of STATES" containing a table with the following columns: FID, Shape\*, AREA, STATE\_NAME, and STATE\_FIPS. The table lists 51 rows of state data, including Alabama, Alaska, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, District of Columbia, Florida, Georgia, Hawaii, Idaho, Illinois, Indiana, Iowa, Kansas, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, and Montana. The bottom of the window features a record navigation bar with buttons for first, previous, next, and last records, a text field showing "0", and a "Show:" dropdown menu set to "All".

| FID | Shape*  | AREA       | STATE_NAME           | STATE_FIPS |
|-----|---------|------------|----------------------|------------|
| 41  | Polygon | 51715.656  | Alabama              | 01         |
| 49  | Polygon | 576556.687 | Alaska               | 02         |
| 35  | Polygon | 113711.523 | Arizona              | 04         |
| 45  | Polygon | 52912.797  | Arkansas             | 05         |
| 23  | Polygon | 157774.187 | California           | 06         |
| 30  | Polygon | 104099.109 | Colorado             | 08         |
| 17  | Polygon | 4976.434   | Connecticut          | 09         |
| 27  | Polygon | 2054.506   | Delaware             | 10         |
| 26  | Polygon | 66.063     | District of Columbia | 11         |
| 47  | Polygon | 55815.051  | Florida              | 12         |
| 43  | Polygon | 58629.195  | Georgia              | 13         |
| 48  | Polygon | 6381.435   | Hawaii               | 15         |
| 7   | Polygon | 83340.594  | Idaho                | 16         |
| 25  | Polygon | 56297.953  | Illinois             | 17         |
| 20  | Polygon | 36399.516  | Indiana              | 18         |
| 12  | Polygon | 56257.219  | Iowa                 | 19         |
| 32  | Polygon | 82195.437  | Kansas               | 20         |
| 31  | Polygon | 40318.777  | Kentucky             | 21         |
| 46  | Polygon | 45835.898  | Louisiana            | 22         |
| 2   | Polygon | 32161.664  | Maine                | 23         |
| 29  | Polygon | 9739.753   | Maryland             | 24         |
| 13  | Polygon | 8172.482   | Massachusetts        | 25         |
| 50  | Polygon | 57898.367  | Michigan             | 26         |
| 9   | Polygon | 84517.469  | Minnesota            | 27         |
| 42  | Polygon | 47618.723  | Mississippi          | 28         |
| 34  | Polygon | 69831.625  | Missouri             | 29         |
| 1   | Polygon | 147236.031 | Montana              | 30         |

Object classes  
which have  
Geometry  
encoded are  
called feature  
classes

# Relation Rules (Codd, 1970)

- Only one value in **each cell** (intersection of row and column)
- All values in a column are about the **same subject**
- Each row is **unique**
- No significance in **column** sequence
- No significance in **row** sequence

# Normalization

- This is the process of converting tables to **conform** to Codd's relational rules
- Split tables into new tables that can be **joined** at query time
  - The **relational join**
- Several levels of normalization
  - Forms: 1NF, 2NF, 3NF, etc.
- Normalization creates many **expensive** joins
- De-normalization is OK for performance optimization

# Relational Join

- We use the **relational join** operation because
  - We are using tables that have been transformed by normalization
  - Data created/maintained by different users, but **integration** needed for queries
  - We want to **combine data** to ask questions that can only be answered by using the data together
- Table joins use **common keys** (column values) filled with the same identifiers
- The table (attribute) join concept has been extended to **geographic cases**

# Relational Join

- Take two tables full of different data (e.g. registrar info & parking data), and **join** them:

| PID #   | Name     | Major | Phone #  | ... |
|---------|----------|-------|----------|-----|
| 1010789 | John D.  | EEOS  | 555-4321 | ... |
| 1021384 | David Q. | Comm. | 555-6789 | ... |

| PID #   | Name     | Parking Lot | License Plate | ... |
|---------|----------|-------------|---------------|-----|
| 1010789 | John D.  | North Lot   | PNT3465       | ... |
| 1021384 | David Q. | Lot 150     | JRS4089       | ... |

The tables are joined through a **common key** which has a unique value for each record










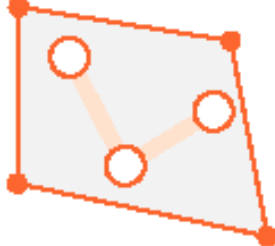

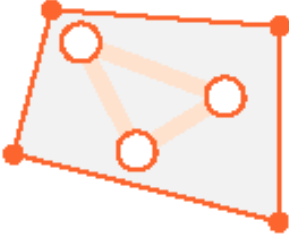


# Spatial Relations

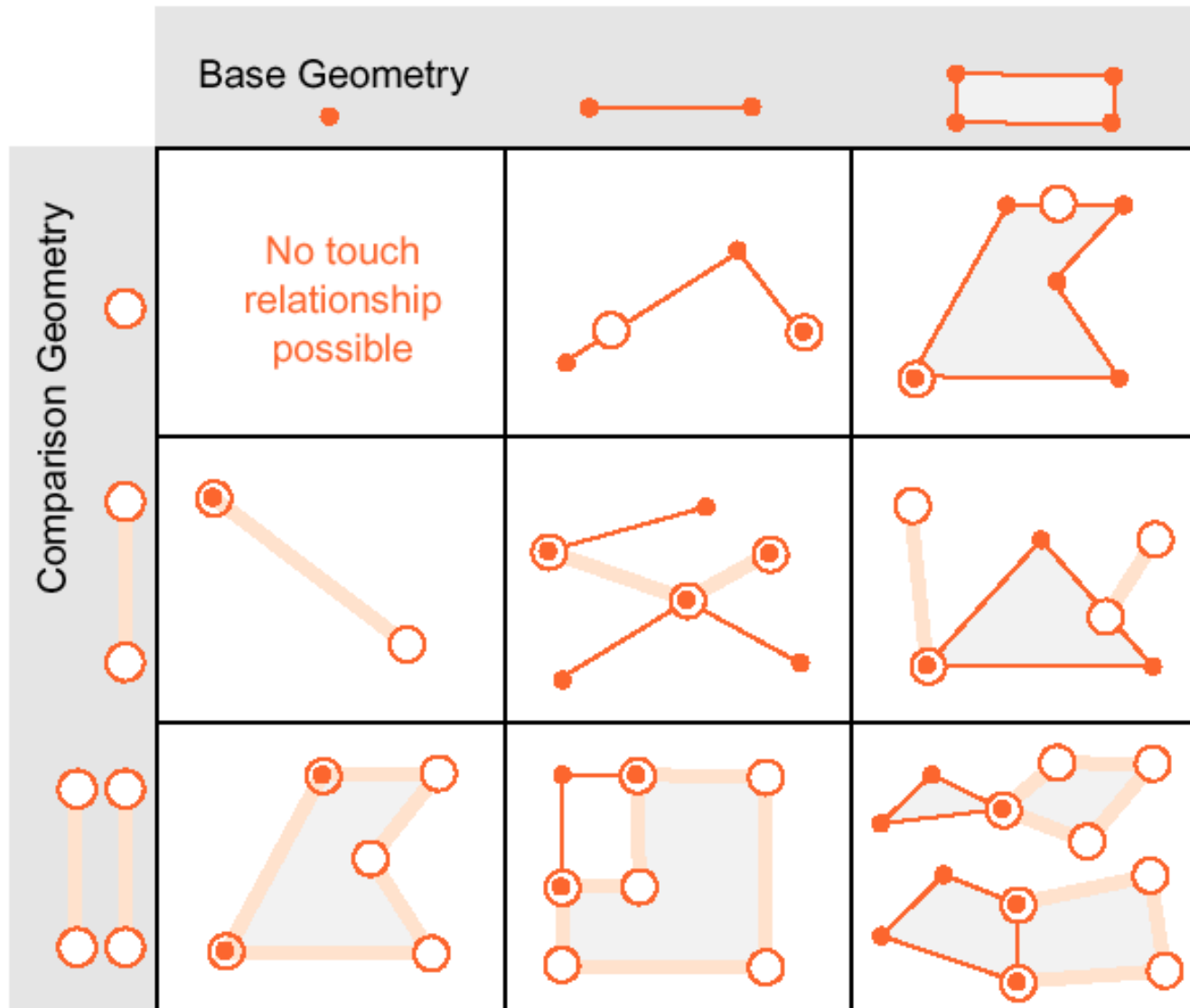
• In addition to relations that join tables based on an **identical common key**, we can evaluate relations between the **spatial characteristics of features**:

- **Equals** – same geometries
- **Disjoint** – geometries share common point
- **Intersects** – geometries intersect
- **Touches** – geometries intersect at common boundary
- **Crosses** – geometries overlap
- **Within** – geometry within
- **Contains** – geometry completely contains
- **Overlaps** – geometries of same dimension overlap
- **Relate** – intersection between interior, boundary or exterior

# Contains Relation

|                     |   | Base Geometry   |   |   |
|---------------------|---|---|---|---|
|                     |   |  |  |    |
| Comparison Geometry |    |  |   |    |
|                     |   | No containment relationship possible  |  |   |
|                     |  | No containment relationship possible  | No containment relationship possible  |  |

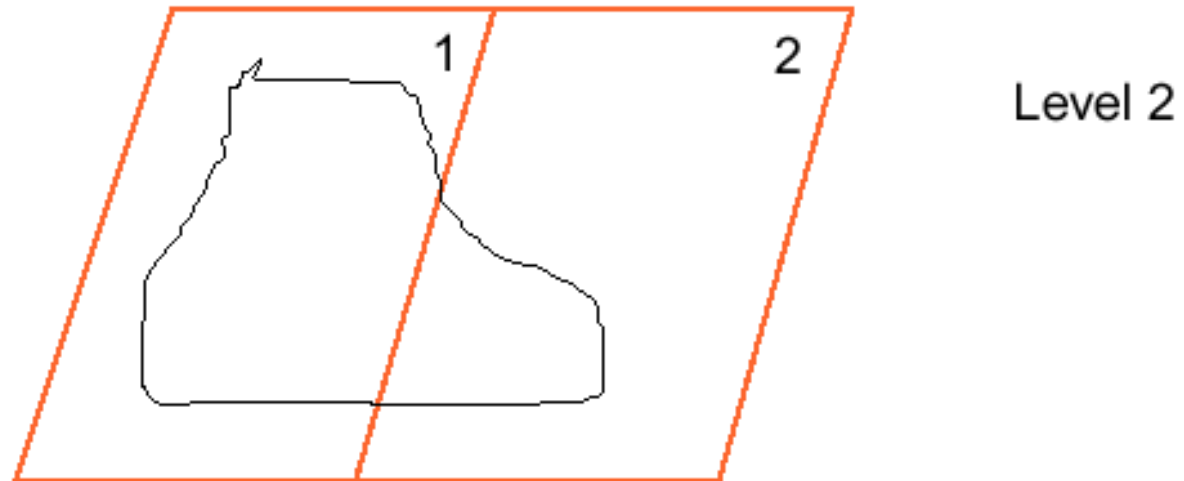
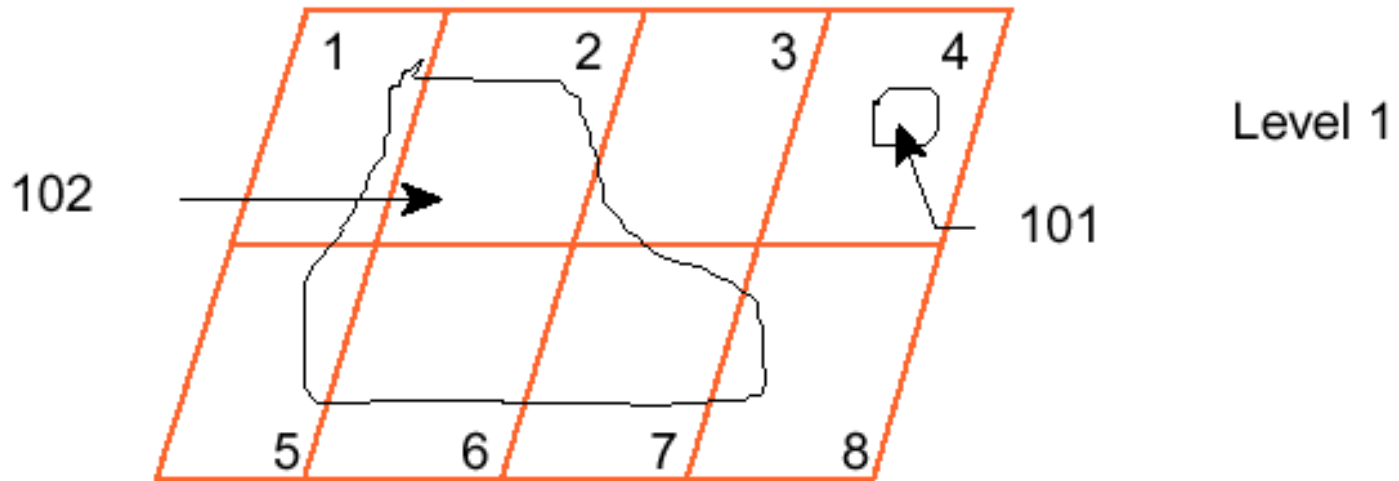
# Touches Relation



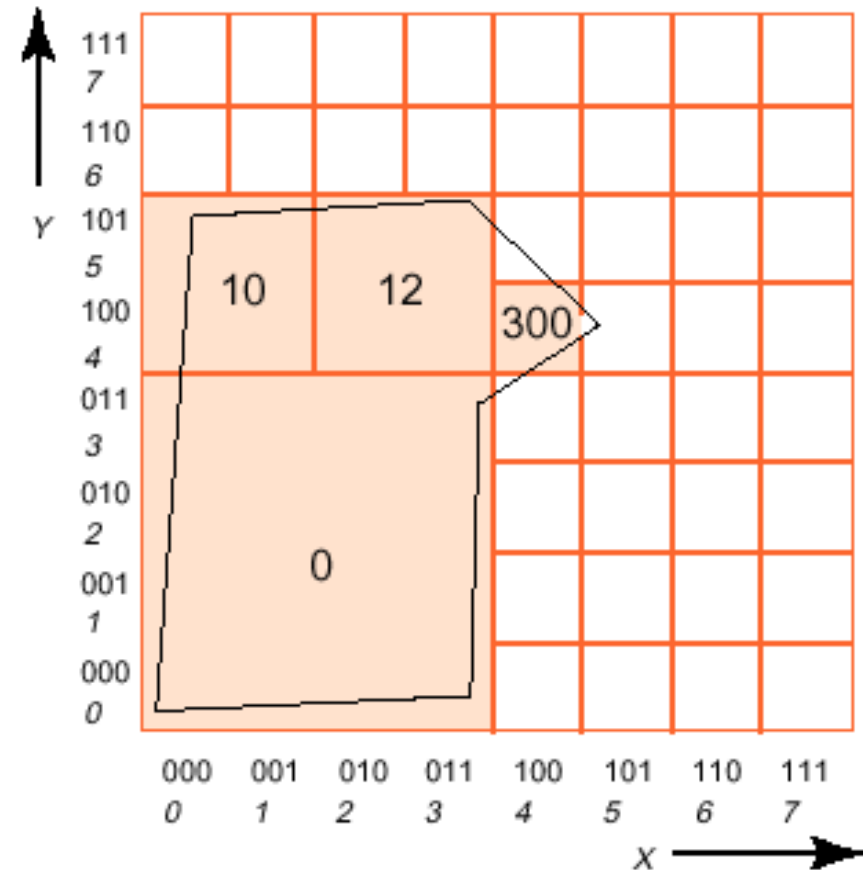
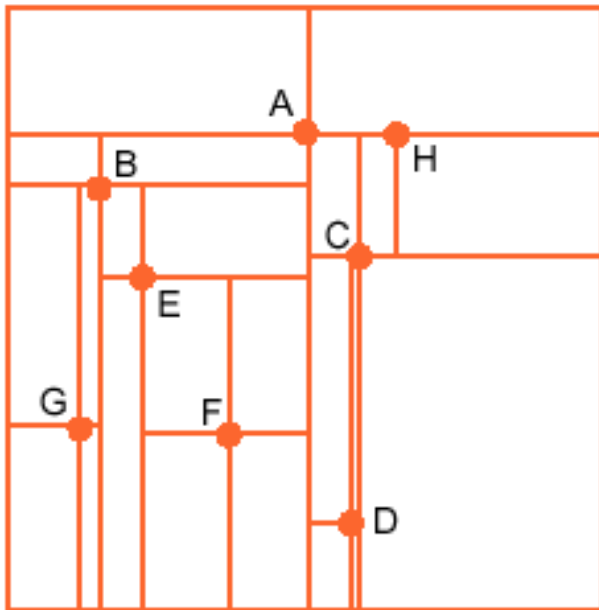
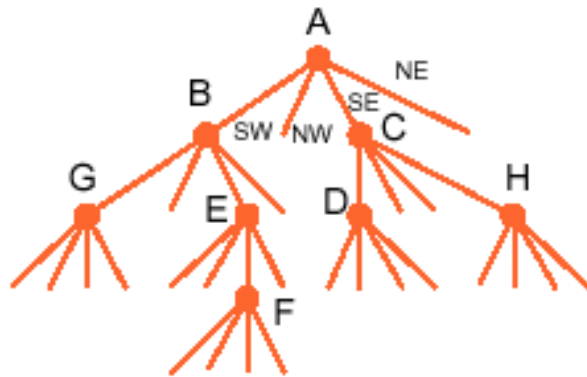
# Indexing

- Used to locate rows **quickly**, speed up access
- RDBMS use **simple 1-d indexing**
- Spatial DBMS need **2-d, hierarchical indexing** to allow features in a given vicinity to be found quickly, using a variety of methods:
  - Grid
  - Quadtree
  - R-tree
  - Others
- Hierarchical in the sense that **multi-level queries** are often used for better performance

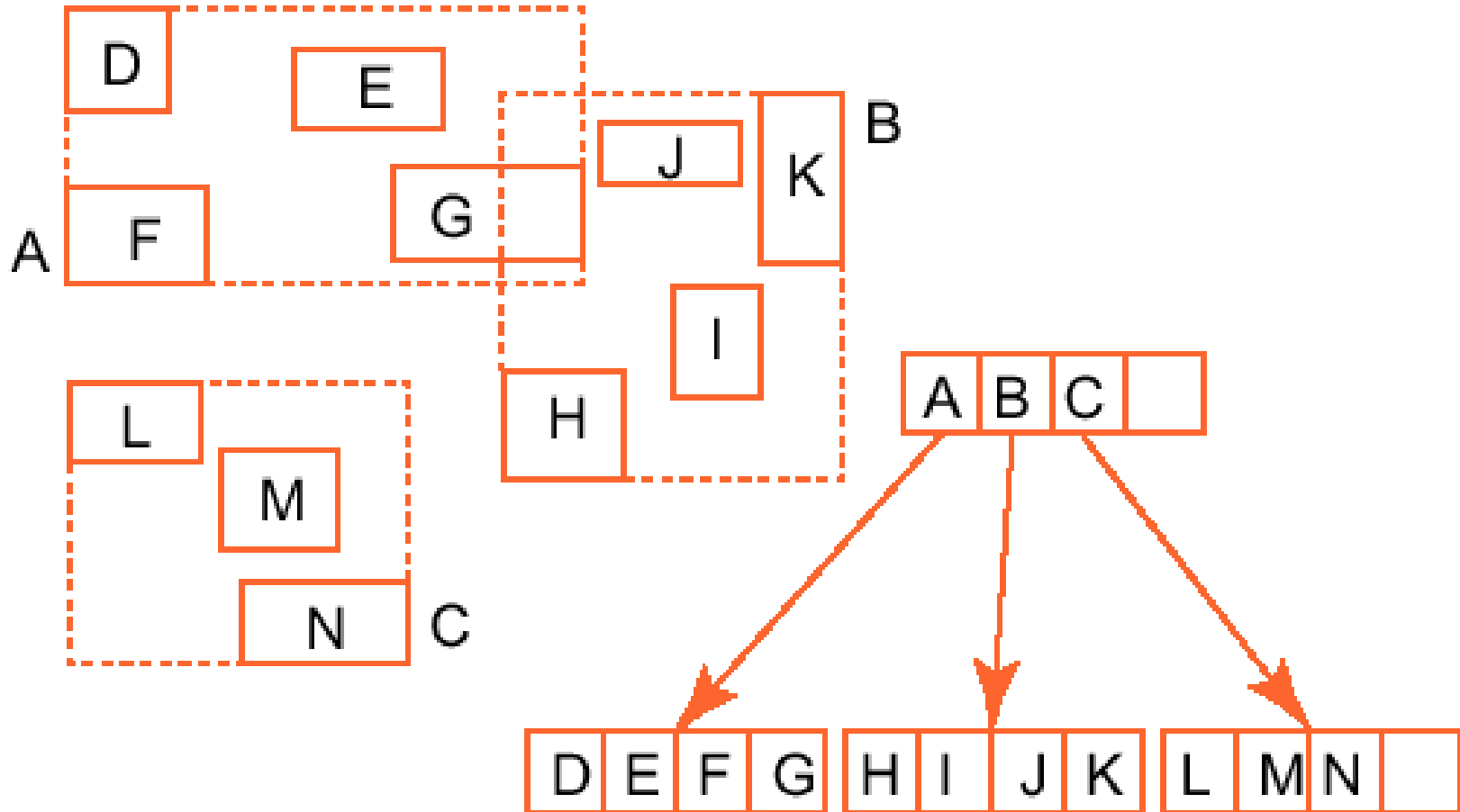
# Grid Index (multi-level)



# Point and Region Quadrees

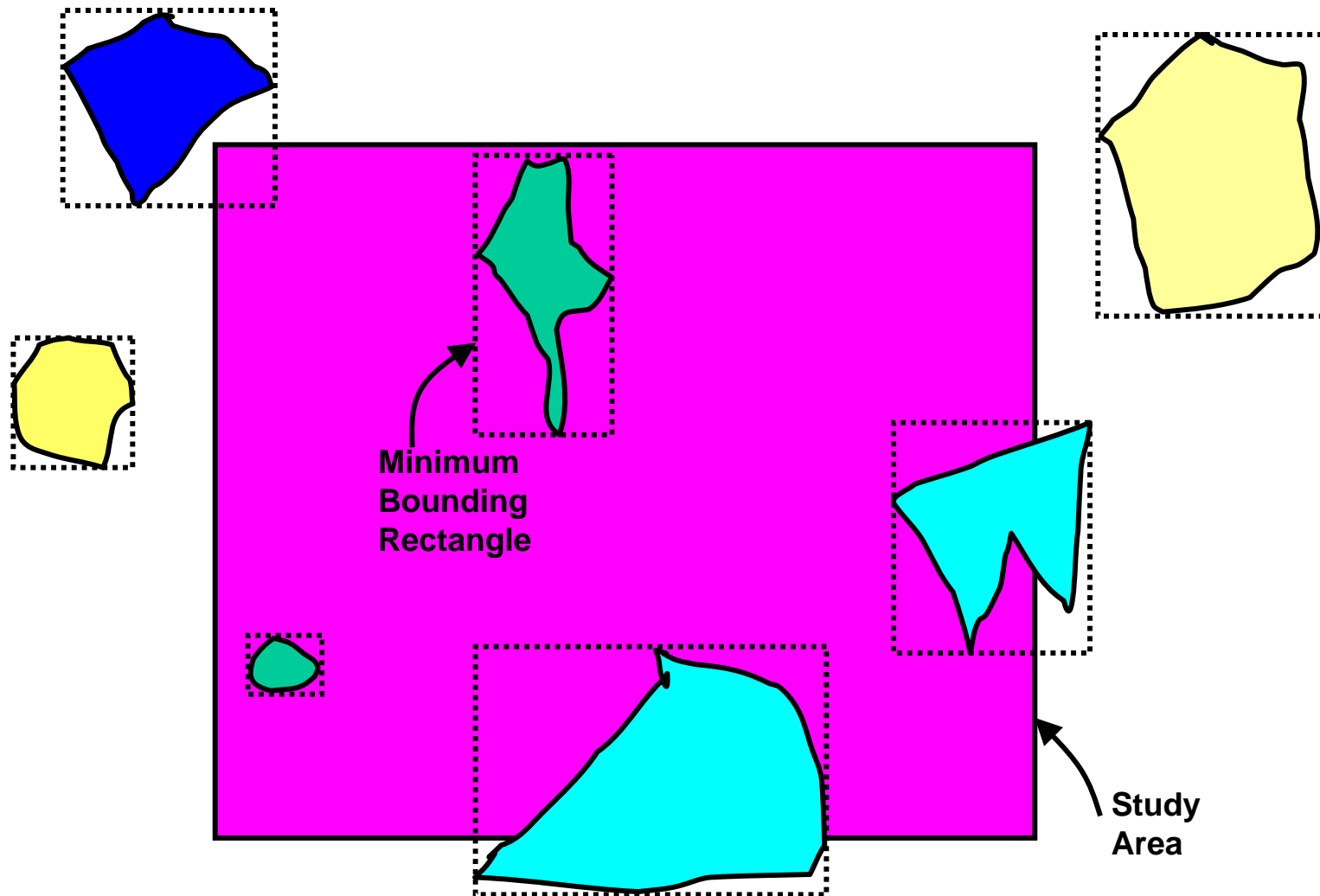


# R-tree



Branching factor  $M = 4$

# Minimum Bounding Rectangle





# Retrieval Operations

- Searches **by attribute**: find and browse.
- Data **reorganization**: select, renumber, and sort.
- Computer allows the creation of **new attributes** based on **calculated values**.

# Command line attribute query

```
find in states where state_name =  
'California'  
<1 record in result>
```

```
use states
```

```
calculate in states population_density =  
population / area  
<50 records in result>
```

```
restrict in states where  
population_density > 1000  
<20 records selected in result>
```

# The Retrieval User Interface

- **GIS query** is usually by command line, batch, menu (GUI) or macro.
- Most GIS packages use the GUI of the computer's operating system to support both a **menu-type query interface** and a **macro or programming language**.
- **SQL** is a standard interface to relational databases and is supported by many GISs.

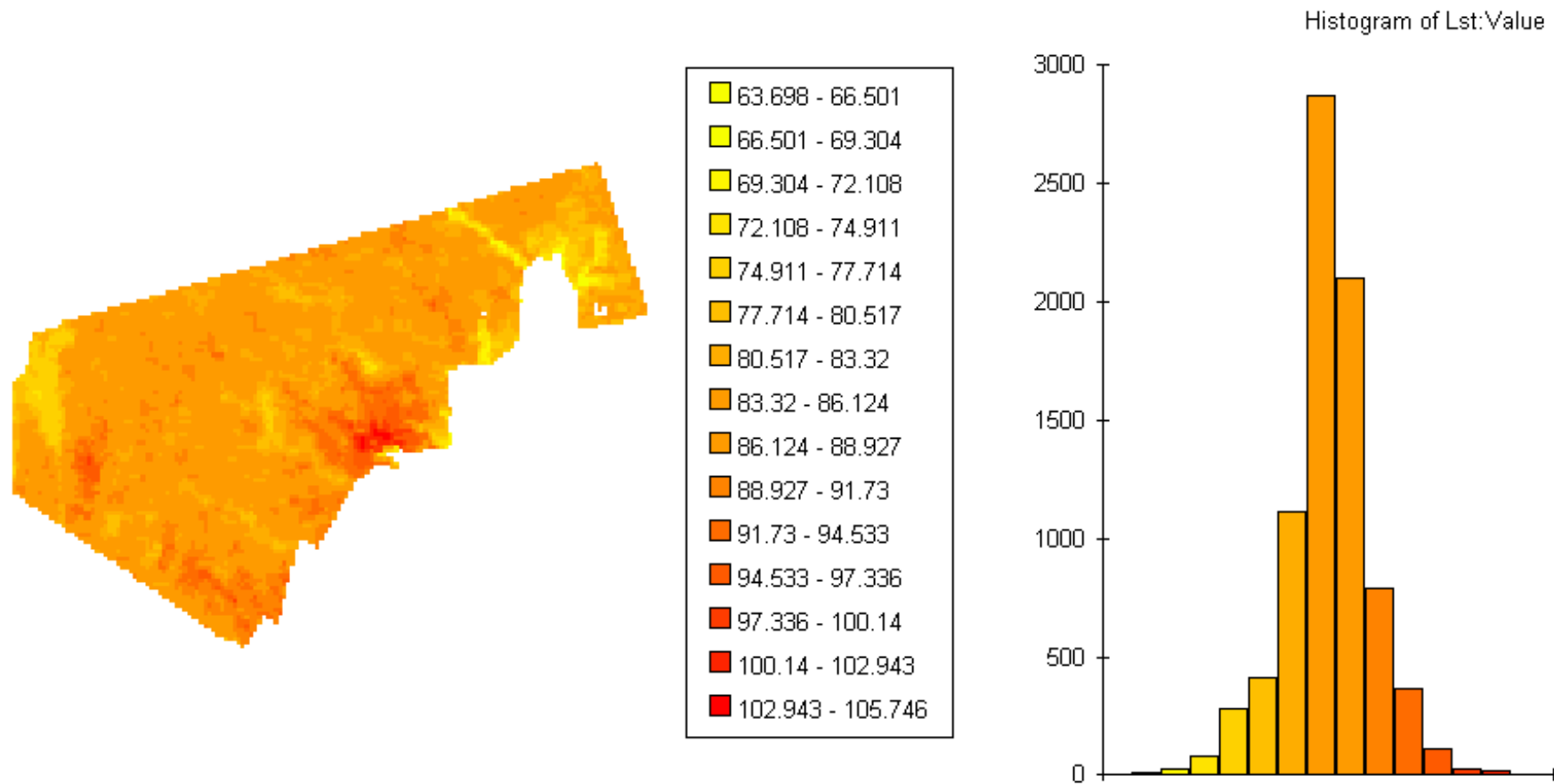
# Spatial Search (Query)

- Identify
- Find
- Buffer
  - a spatial retrieval **around** points, lines, or areas based on distance.
- Near
- Point Distance
- Overlay: Erase, Identity, Intersect, Symmetrical Difference, Union, Update
  - a spatial retrieval operation that is **equivalent to an attribute join**.

# Queries and Reasoning

- A basic function of GIS is the ability to **query** data layers
- Queries can be **attribute-based** (e.g. show me all the pixels in a GRID with an LST value  $> 80$  degrees) or **location-based** (e.g. find all the counties in Maryland Climate Division 6 that are adjacent to Baltimore County)
- A GIS can respond to queries by presenting data in appropriate documents (e.g. **Data View** and/or **a Table**)
- It is often useful to be able to **display two or more documents** at once
  - If we have performed a query that selects features from a theme based on some criteria, we can see the selected features in **both** a Data View and a Table

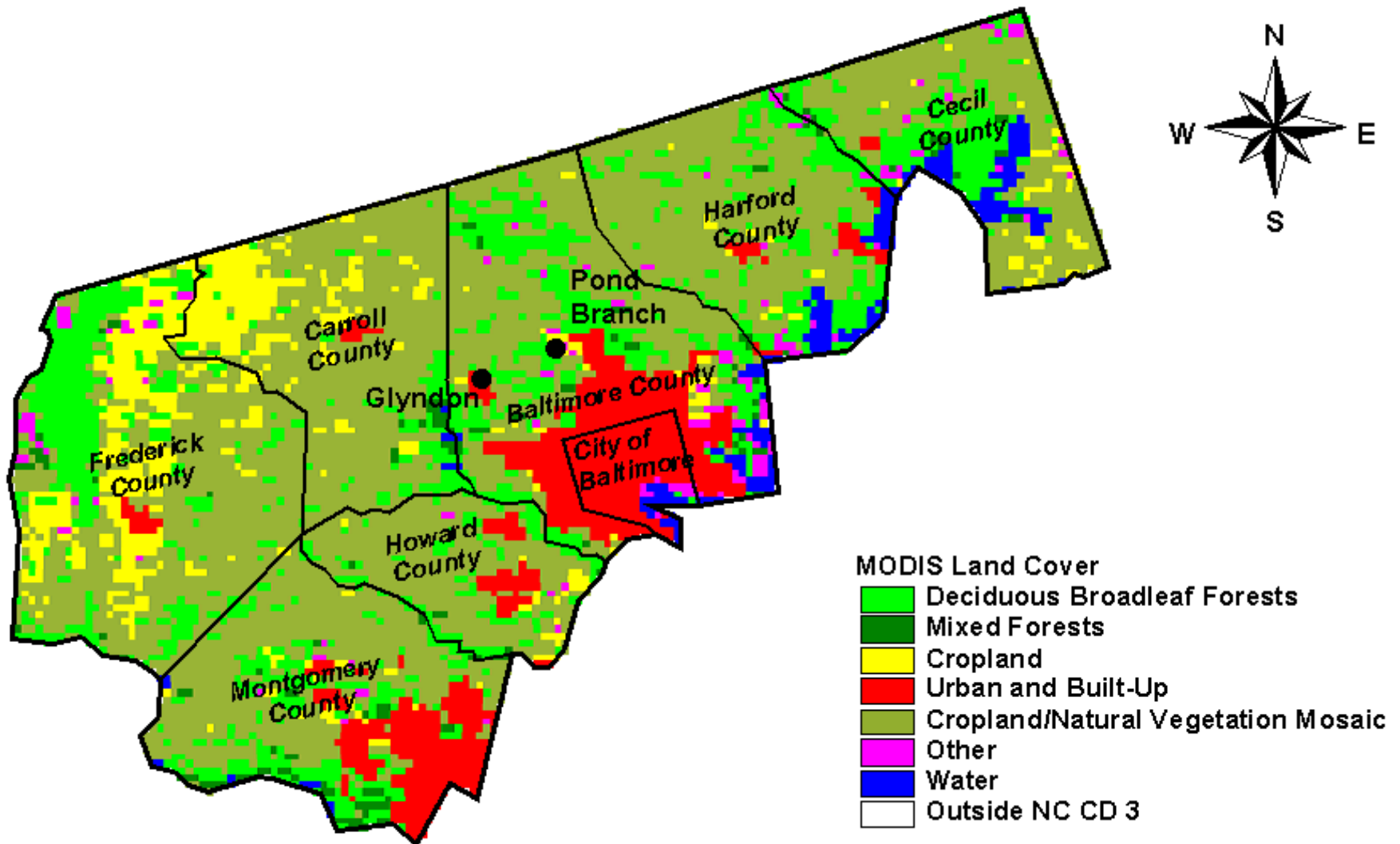
# Land Surface Temperature Example



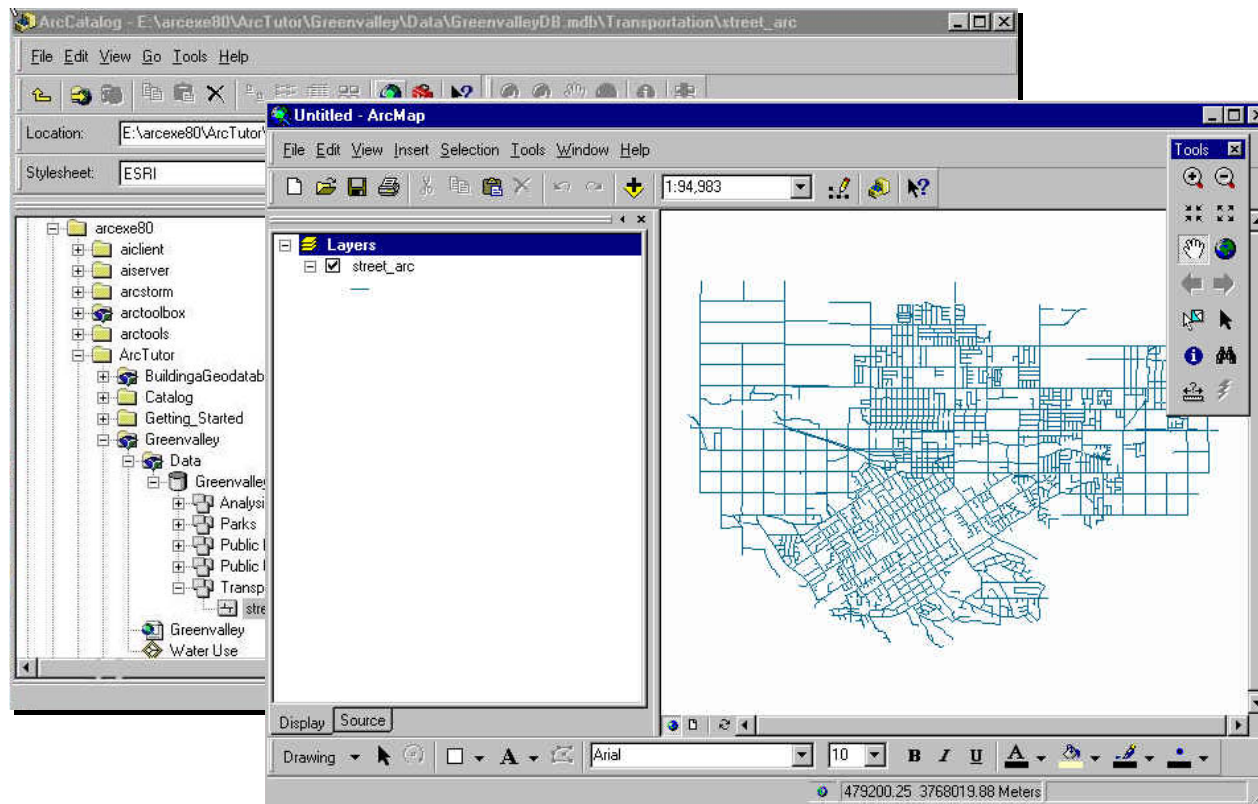
Mean = 85.4994 degrees F

Standard Deviation = 4.3092 degrees F

# MD Climate Division 6 Example



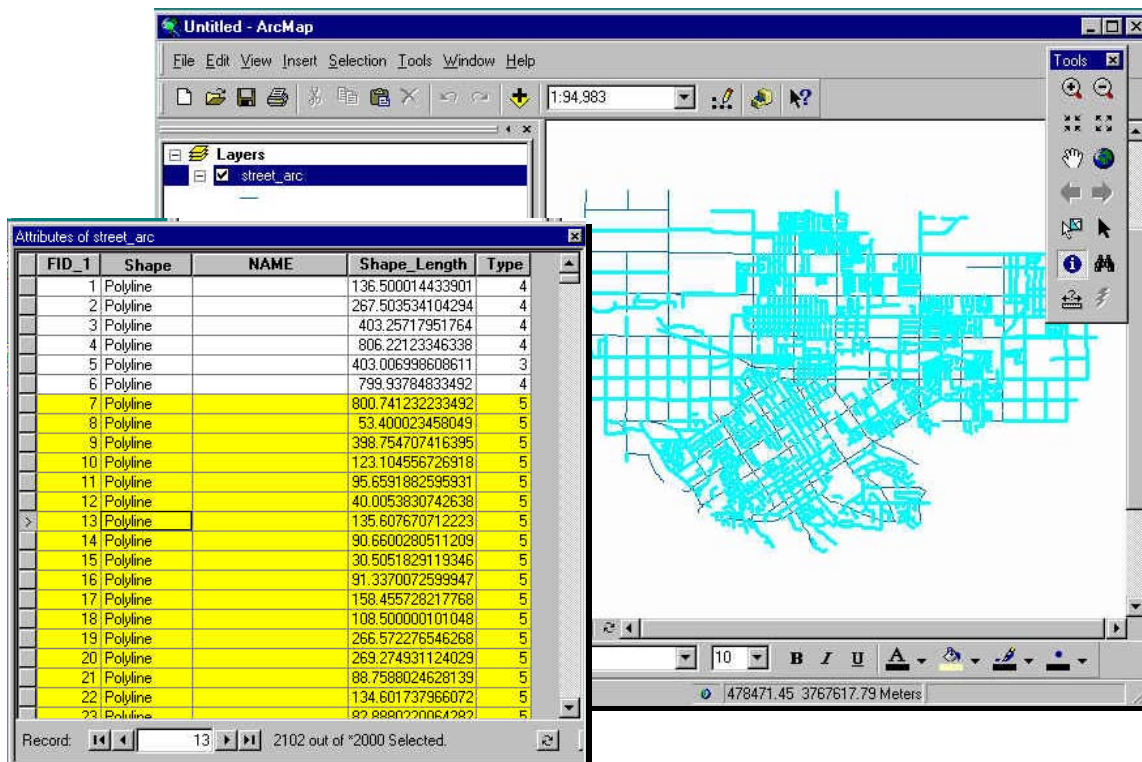
# The Map View (~ Data View)



- A user can interact with a **map view** to identify objects and query their attributes, to search for objects meeting specified criteria, or to find the coordinates of objects. This illustration uses ESRI's ArcMap.



# The Table View (~ a Table)



The screenshot shows the ArcMap interface with a table view of street attributes. The table has the following columns: FID\_1, Shape, NAME, Shape\_Length, and Type. The data is as follows:

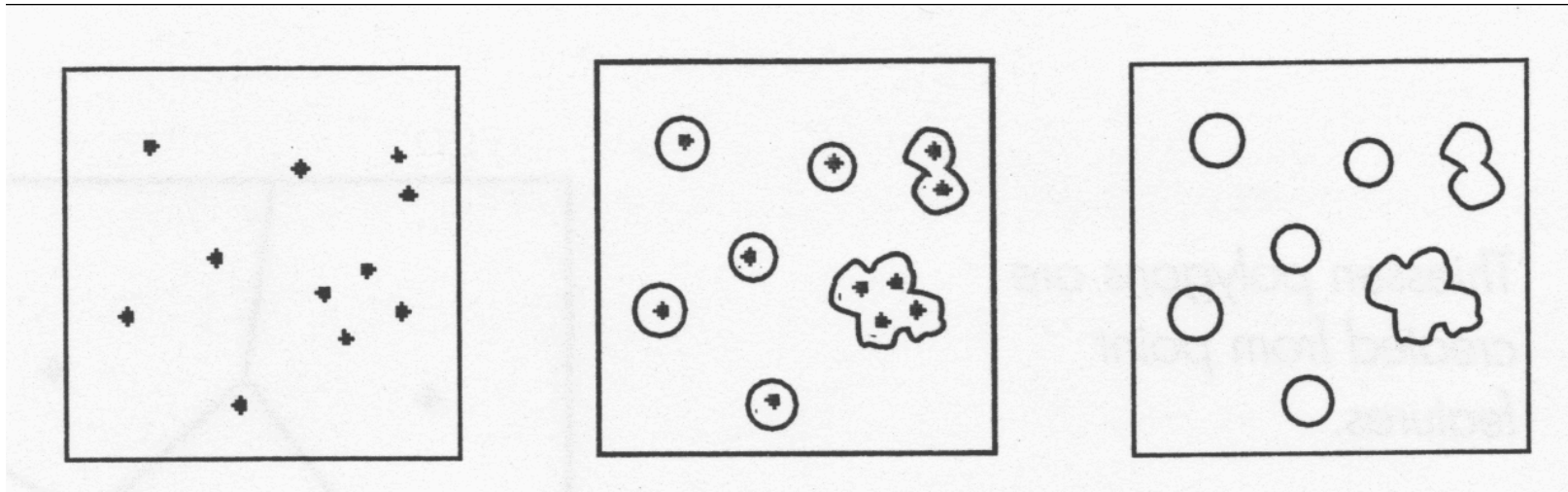
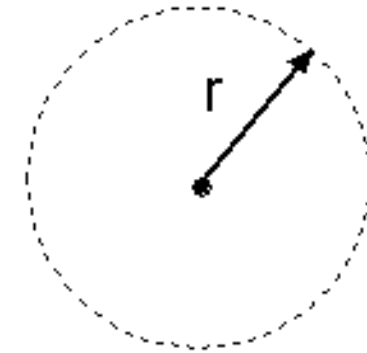
| FID_1 | Shape    | NAME | Shape_Length     | Type |
|-------|----------|------|------------------|------|
| 1     | Polyline |      | 136.500014433901 | 4    |
| 2     | Polyline |      | 267.503534104294 | 4    |
| 3     | Polyline |      | 403.25717951764  | 4    |
| 4     | Polyline |      | 806.22123346338  | 4    |
| 5     | Polyline |      | 403.006998608611 | 3    |
| 6     | Polyline |      | 799.93784833492  | 4    |
| 7     | Polyline |      | 800.741232233492 | 5    |
| 8     | Polyline |      | 53.400023458049  | 5    |
| 9     | Polyline |      | 398.754707416395 | 5    |
| 10    | Polyline |      | 123.104556726918 | 5    |
| 11    | Polyline |      | 95.6591882595931 | 5    |
| 12    | Polyline |      | 40.0053830742638 | 5    |
| 13    | Polyline |      | 135.607670712223 | 5    |
| 14    | Polyline |      | 90.6600280511209 | 5    |
| 15    | Polyline |      | 30.5051829119346 | 5    |
| 16    | Polyline |      | 91.3370072599947 | 5    |
| 17    | Polyline |      | 158.455728217768 | 5    |
| 18    | Polyline |      | 108.500000101048 | 5    |
| 19    | Polyline |      | 266.572276546268 | 5    |
| 20    | Polyline |      | 269.274931124029 | 5    |
| 21    | Polyline |      | 88.7588024628139 | 5    |
| 22    | Polyline |      | 134.601737966072 | 5    |
| 23    | Polyline |      | 87.888022064282  | 5    |

The map view shows a street network with several streets highlighted in cyan, corresponding to the selected rows in the table. The status bar at the bottom indicates the current record is 13 out of 2102 selected records.

• Here attributes are displayed in the form of **a table**, linked to a map view. When objects are selected in the table, they are automatically highlighted in the map view, and vice versa. The table view can be used to answer simple queries about objects and their attributes.

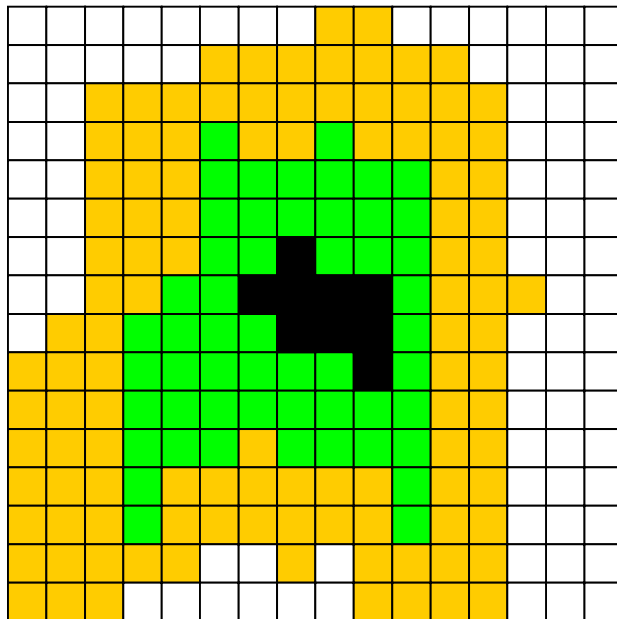
# Buffering (Proximity Analysis)

**Buffering:** The delineation of a zone around the feature of interest **within a given distance**. For a point feature, it is simply a circle with its radius equal to the buffer distance:



# Raster Buffering

- Buffering operations also can be performed using the **raster data model**, where the distance is expressed in terms of the number of adjacent cells included
- In the raster model, we can vary the distance buffered according to values in a **friction** layer (e.g. travel time):

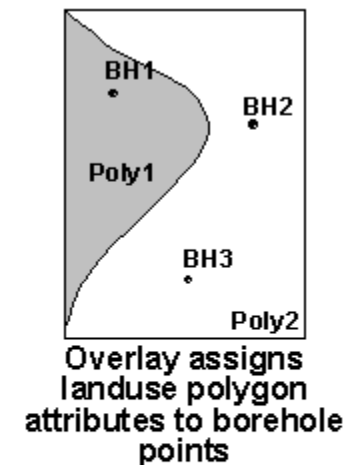
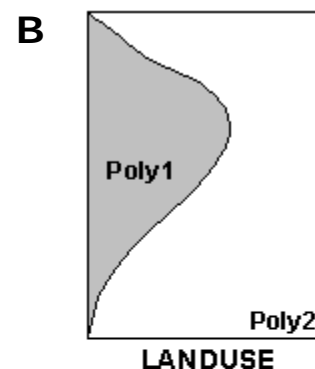
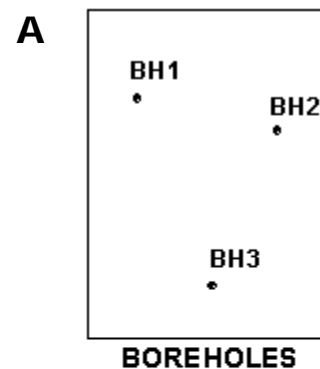


- City limits
- Areas reachable in 5 minutes
- Areas reachable in 10 minutes
- Other areas

# Point in Polygon Analysis

- Overlay point layer (A) with polygon layer (B)
  - **In which** B polygon are A points located?
  - » **Assign polygon attributes** from B to points in A

**Example:** Comparing soil mineral content at sample borehole locations (points) with land use (polygons)...



| Point | Zn  | Pb |
|-------|-----|----|
| BH1   | 140 | 65 |
| BH2   | 178 | 54 |
| BH3   | 101 | 87 |

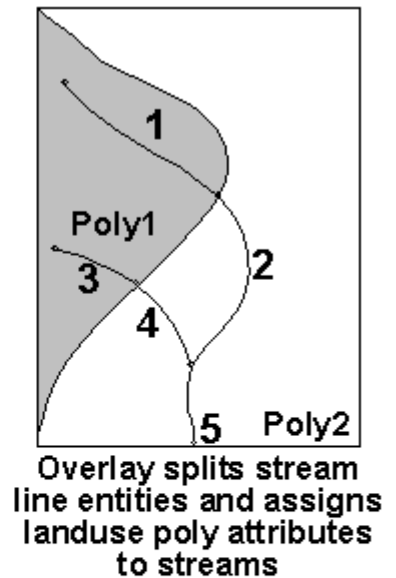
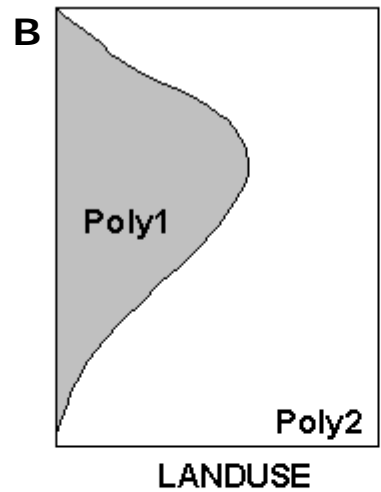
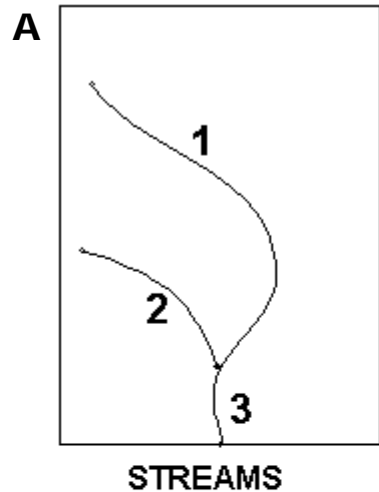
| Poly | Landuse     |
|------|-------------|
| 1    | Agriculture |
| 2    | Urban       |

| Point | Zn  | Pb | Landuse     |
|-------|-----|----|-------------|
| BH1   | 140 | 65 | Agriculture |
| BH2   | 178 | 54 | Urban       |
| BH3   | 101 | 87 | Urban       |

# Line in Polygon Analysis

- Overlay line layer (A) with polygon layer (B)
  - **In which** B polygons are A lines located?
  - » **Assign polygon attributes** from B to lines in A

**Example:**  
Assign land use attributes (polygons) to streams (lines):

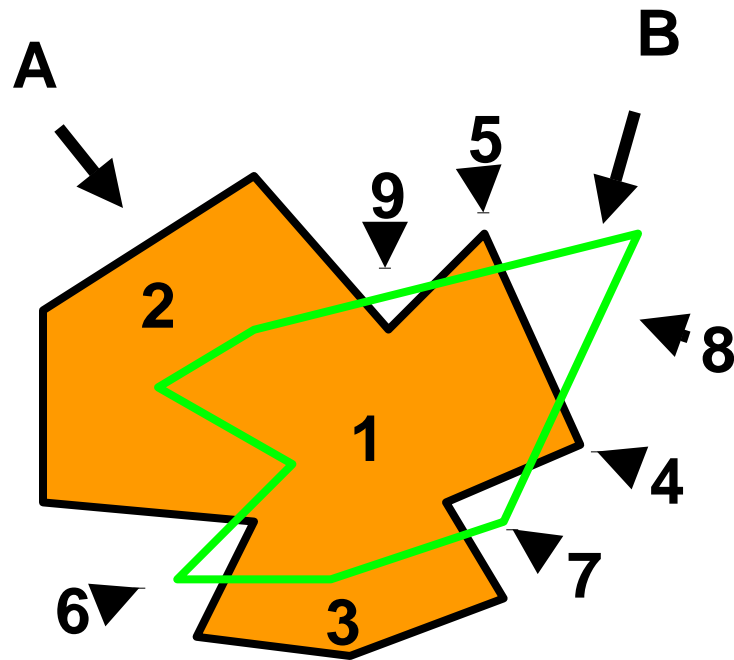


| Line | Length |
|------|--------|
| 1    | 780    |
| 2    | 520    |
| 3    | 225    |

| Poly | Landuse     |
|------|-------------|
| 1    | Agriculture |
| 2    | Urban       |

| Line | Length | Landuse     |
|------|--------|-------------|
| 1    | 440    | Agriculture |
| 2    | 340    | Urban       |
| 3    | 220    | Agriculture |
| 4    | 300    | Urban       |
| 5    | 225    | Urban       |

# Polygon Overlay, Discrete Object Case

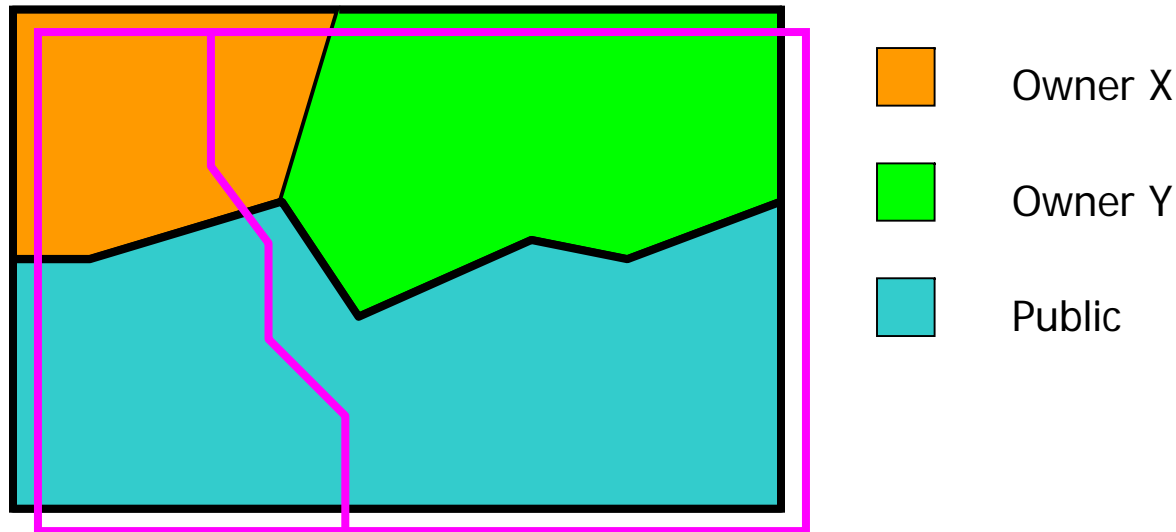


- In this example, the **union** of two polygons is taken to form **nine** new polygons. One is formed from both input polygons (1); four are formed by Polygon A and not Polygon B (2-5); and four are formed by Polygon B and not Polygon A (6-9)

# Polygon Overlay, Field Case

- **Two layers** of edge-to-edge polygons are the inputs, representing two thematic descriptions of the same area, e.g. soil type and land ownership information
- The **two layers are overlaid**, and all intersections are computed, creating a new layer:
  - Each polygon in the new layer has **both** a soil type and land ownership information
  - The two attributes are said to be **concatenated**
- The task is often performed using the **raster** spatial data model, but can use vector map algebra

# Polygon Overlay, Field Case



- A layer representing a field of **land ownership** (symbolized using colors) is overlaid on a **layer of soil type** (layers offset for emphasis). The result after overlay will be a single layer with **5 polygons**, each with land ownership information and a soil type



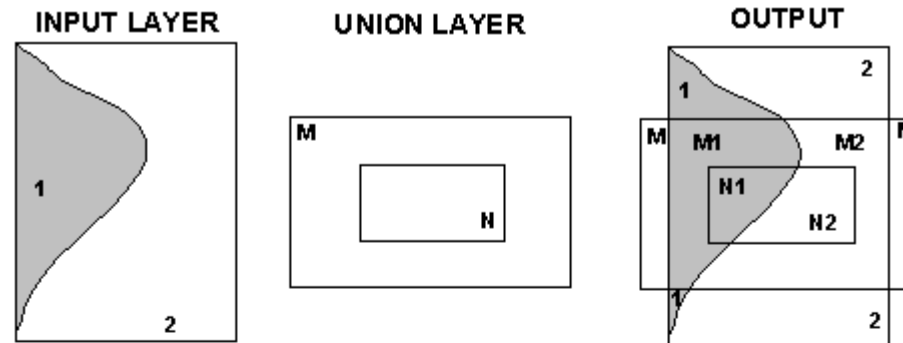
# Polygon Overlay Analysis

- Overlay polygon layer (A) with polygon layer (B)
  - **What** are the **spatial polygon combinations** of A and B?
    - » Generate a **new data layer** with **combined polygons**
      - **attributes from both** layers are included in output
- **How** are polygons combined (i.e. what geometric rules are used for combination)?
  - UNION (Boolean OR)
  - INTERSECTION (Boolean AND)
  - IDENTITY
- Polygon overlay will generally result in a **significant increase** in the **number of spatial entities** in the output
  - can result in output that is **too complex** to interpret

# Polygon Overlay Analysis

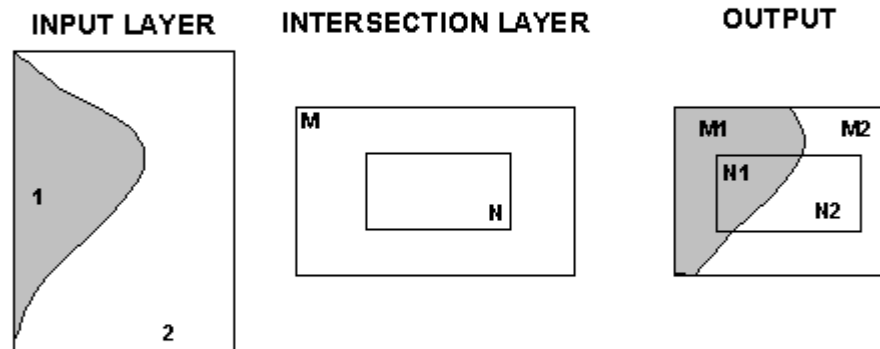
## UNION

- overlay polygons and **keep areas from both layers**



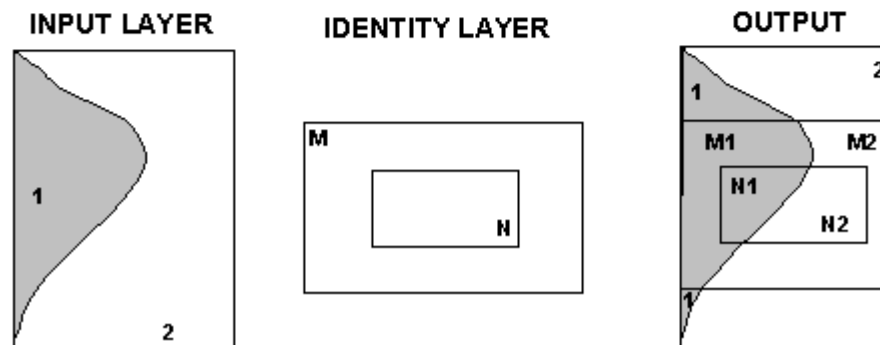
## INTERSECTION

- overlay polygons and **keep only areas in the input layer that fall within the intersection layer**



## IDENTITY

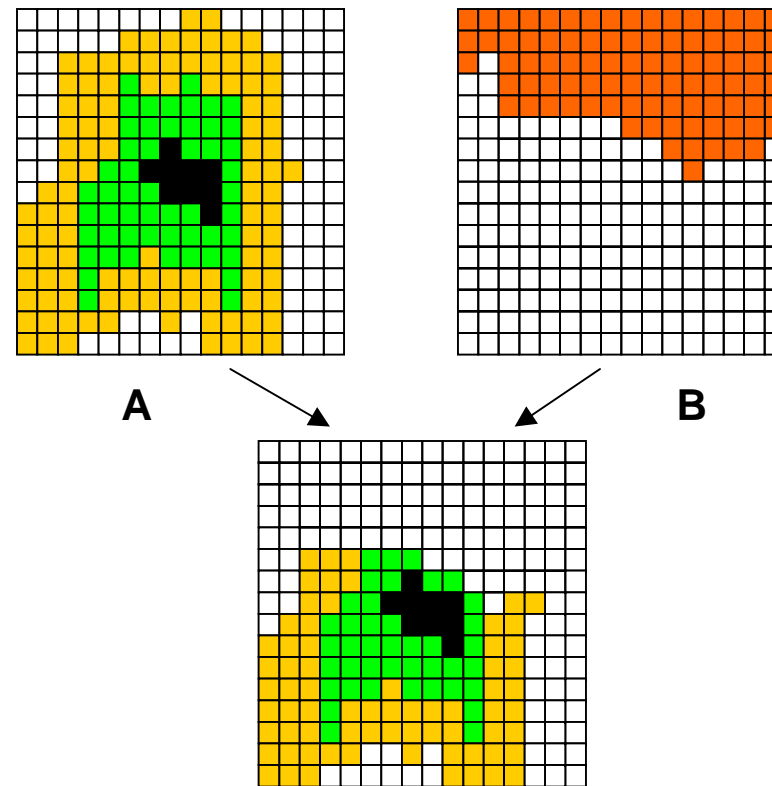
- overlay polygons and **keep areas from input layer**



# Complex Retrieval: Map Algebra

- **Combinations** of spatial and attribute queries can build some **complex and powerful** GIS operations, such as weighting.
- Weighted overlay analysis really just **complex retrieval**.

# Overlay of Fields Represented as Rasters



The two input data sets are maps of (A) travel time from the urban area shown in black, and (B) county (red indicates County X, white indicates County Y). The output map identifies **travel time to areas in County Y only**, and might be used to compute average travel time to points in that county in a subsequent step

# Algebraic Operations w/ Raster Layers

- Map algebra:
  - Treats input layers as **numeric inputs** to mathematical operations (each layer is a separate numeric input)
  - The result of the operation on the inputs is calculated on a **cell-by-cell basis**
- This allows for **complex overlay analyses** that can use as many input layers and operations as necessary
- A common application of this approach is **suitability analysis** where multiple input layers determine suitable sites for a desired purpose by **scoring cells** in the input layers according to their effect on suitability and combining them, often **weighting layers** based on their importance

# Simple Arithmetic Operations

## Summation

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 1 & 1 & 2 \\ \hline 1 & 0 & 2 \\ \hline \end{array}$$

## Multiplication

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

## Summation of more than two layers

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 1 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 1 & 1 \\ \hline 2 & 2 & 3 \\ \hline 1 & 0 & 3 \\ \hline \end{array}$$

# Raster (Image) Difference

The difference between two layers

$$\begin{array}{|c|c|c|} \hline 5 & 4 & 3 \\ \hline 6 & 5 & 6 \\ \hline 7 & 1 & 5 \\ \hline \end{array} - \begin{array}{|c|c|c|} \hline 3 & 5 & 6 \\ \hline 1 & 4 & 5 \\ \hline 3 & 2 & 7 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 2 & -1 & -3 \\ \hline 5 & 1 & 1 \\ \hline 4 & -1 & -2 \\ \hline \end{array}$$

- An application of taking the differences between layers is **change detection**:
  - Suppose we have **two raster layers** that each show a map of the **same phenomenon** at a particular location, and each was generated at a **different point in time**
  - By taking the **difference** between the layers, we can **detect changes** in that phenomenon over that interval of time
- Question: **How** can the locations where changes have occurred be identified using the difference layer?

# More Complex Operations

## Linear Transformation

$$\mathbf{a} \begin{array}{|c|c|c|} \hline 1 & 2 & 4 \\ \hline 3 & 2 & 1 \\ \hline 5 & 3 & 2 \\ \hline \end{array} + \mathbf{b} \begin{array}{|c|c|c|} \hline 1 & 0 & 0 \\ \hline 5 & 1 & 1 \\ \hline 2 & 0 & 1 \\ \hline \end{array} + \mathbf{c} \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

- We can multiply layers by **constants** (such as a, b, and c in the example above) before summation
- This could be applied in the context of computing the results of a **regression model** (e.g. output  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ ) using raster layers
- Another application is **suitability analysis**, where individual **input layers** might be **various criteria**, and the **constants** a, b, and c determine the **weights** associated with those criteria



# Chapter 5: What is Where?

- 5.1 Basic Database Management
- 5.2 Searches By Attribute
- 5.3 Searches By Geography
- 5.4 The Query Interface

# Next Topic:

Why is it there?