Chapter 4. Attribute Data

Objectives:

- Understanding how **tabular data** are stored and used
- Understanding the links between **database management systems** and **tables**
- (Using **queries** to select **records of interest**)
- Understanding **joins** and **cardinality** concepts
- **Summarizing** tables to get **statistics** on groups
- Learning how to **define fields**
- **Editing** and **calculating fields** in tables

A GIS can answer the question: What is where?

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

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

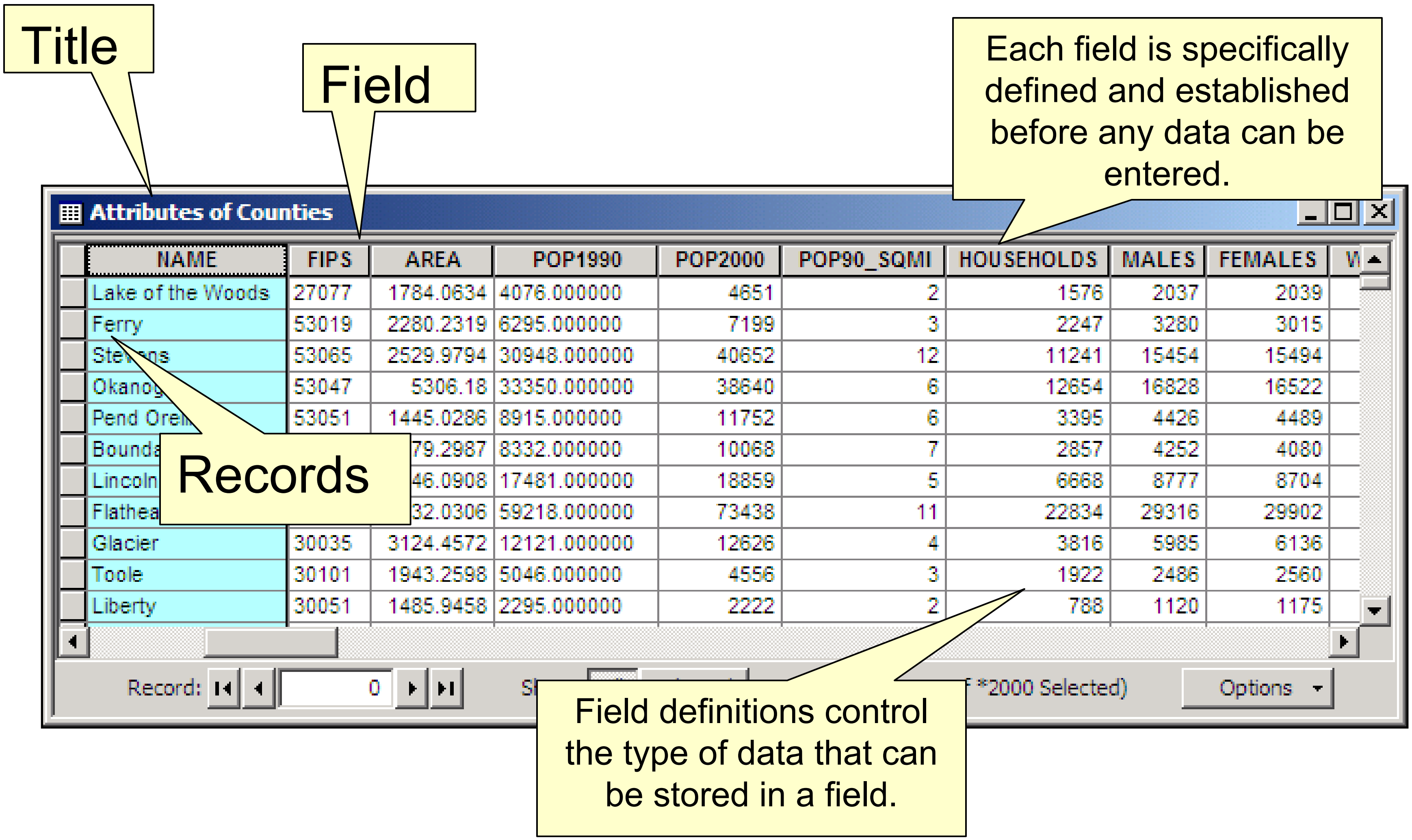
Spatial data: Describing **where** things are

<u>AND</u>

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

Table terminology



NAME	FIPS	AREA	POP1990	POP2000	POP90_SQMI	HOUSEHOLDS	MALES	FEMALES
Lake of the Woods	27077	1784.0634	4076.000000	4651	2	1576	2037	2039
Ferry	53019	2280.2319	6295.000000	7199	3	2247	3280	3015
Ste..ns	53065	2529.9794	30948.000000	40652	12	11241	15454	15494
Okano..	53047	5306.18	33350.000000	38640	6	12654	16828	16522
Pend Ore..	53051	1445.0286	8915.000000	11752	6	3395	4426	4489
Bounda..		..79.2987	8332.000000	10068	7	2857	4252	4080
Lincoln		..46.0908	17481.000000	18859	5	6668	8777	8704
Flathea..		..32.0306	59218.000000	73438	11	22834	29316	29902
Glacier	30035	3124.4572	12121.000000	12626	4	3816	5985	6136
Toole	30101	1943.2598	5046.000000	4556	3	1922	2486	2560
Liberty	30051	1485.9458	2295.000000	2222	2	788	1120	1175

Types of tables

- **Attribute table**
  - Stores **attributes** of **map features**
  - **Associated** with a **spatial data layer**
  - Has **special fields** for **spatial information**

- **Standalone table**
  - Stores **any tabular data**
  - **Not associated** with **spatial data**
  - **OID** instead of **FID**

Attributes of US Counties

FID	Shape*	NAME	STATE_NAME
0	Polygon	Lake of the Woods	Minnesota
1	Polygon	Ferry	Washington
2	Polygon	Stevens	Washington
3	Polygon	Okanogan	Washington
4	Polygon	Pend Oreille	Washington

Attributes of popestmt

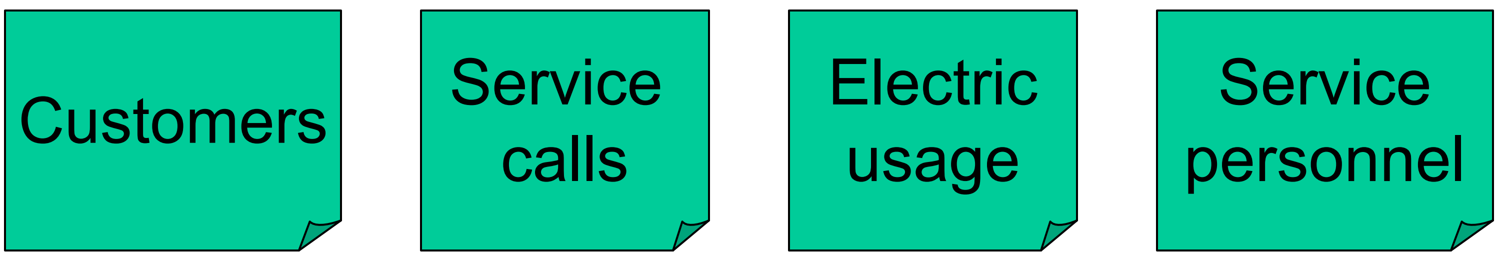
OID	FIPS	POP1998	POP1997	POP1996
0	01001	42095	41284	40251
1	01003	132828	128820	124257
2	01005	26895	26791	26870
3	01007	18926	18595	18227
4	01009	46266	44930	43548

Database Management Systems

- **Dedicated systems** for **managing tables** of data
- Provide **data management** for agencies, universities, companies, etc.
- Designed for **multi-user environments** with **enhanced security needs**
- **Focus on data tables** with **tools** for queries, reporting, graphing, etc.

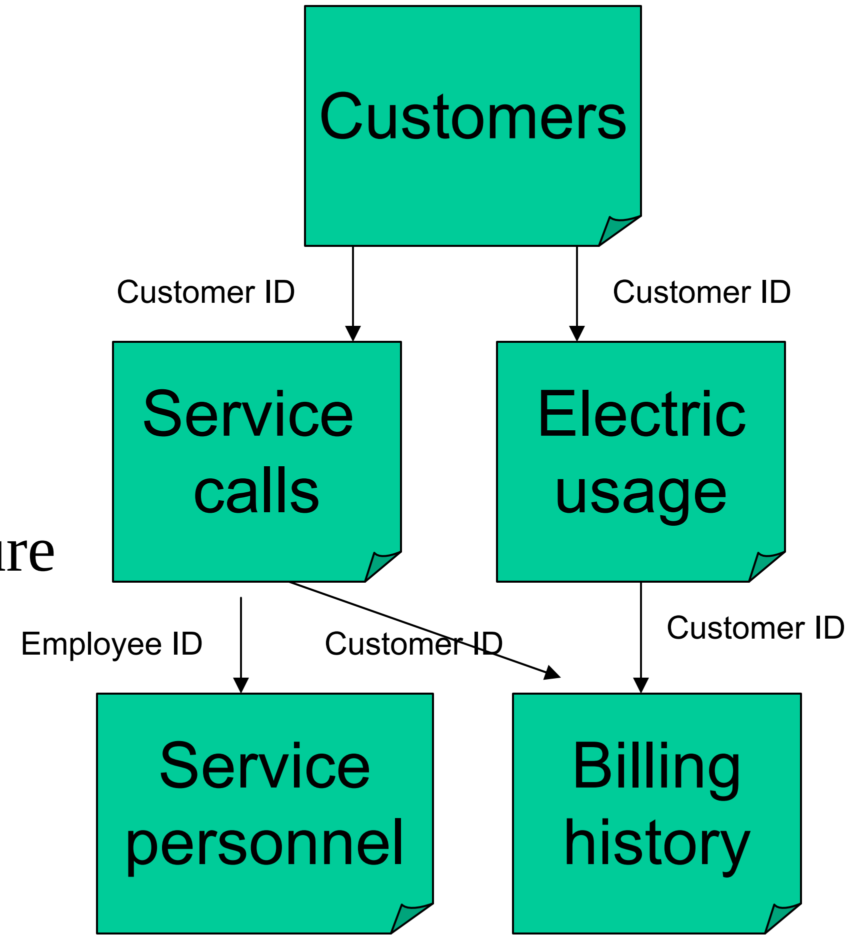
Flat file DBMS

- **Flat file**
  - Stores data as **rows of information in files**
  - **Simple** and robust
  - **Inefficient** for search and query



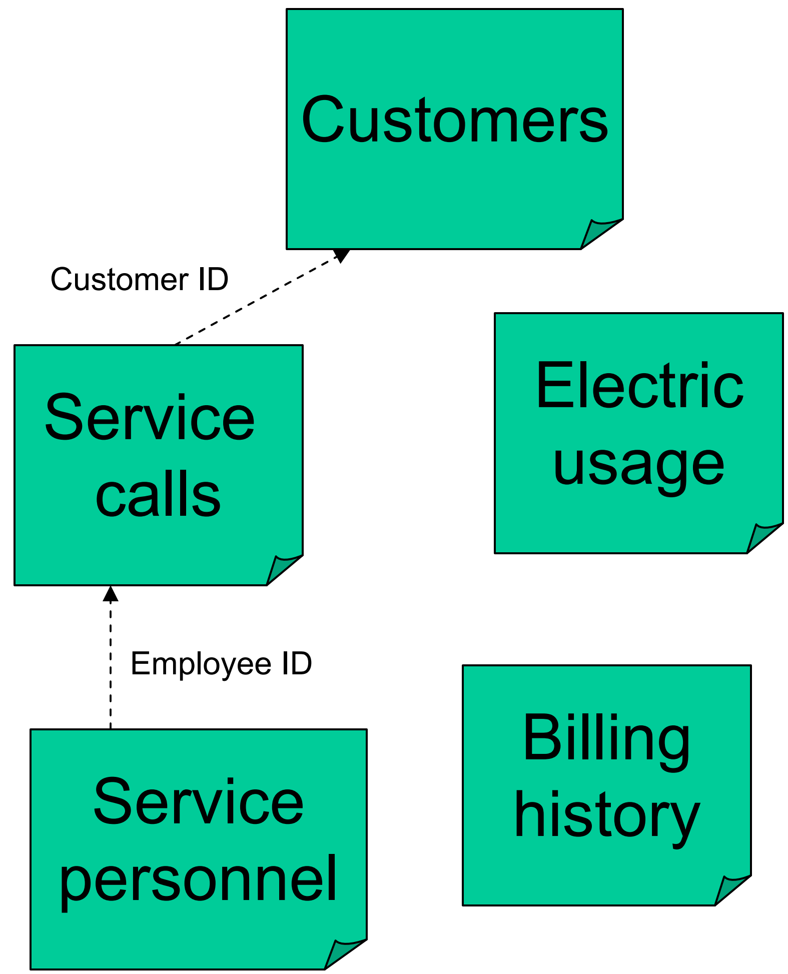
Hierarchical DBMS

- Stores data in **multiple tables**
- Tables have **defined parent-child relationships**
- **Pre-set hierarchy** of table relationships **designed for specific queries**
- **Very efficient** for **specific queries**
- Range of queries **limited** by structure

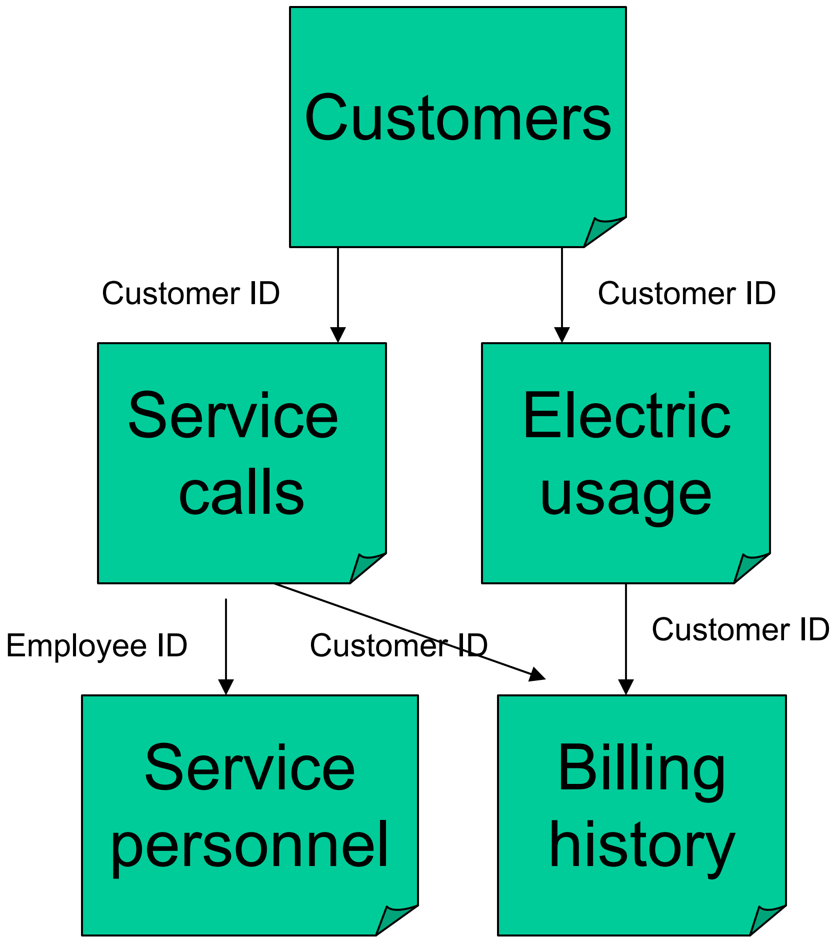


Relational DBMS

- Stores data in **multiple tables**
- Table relationships are **defined as needed**
- **Very flexible**
- Ideal for **open-ended applications** when queries not known beforehand
- **Most common type** used in **GIS applications**



Hierarchical

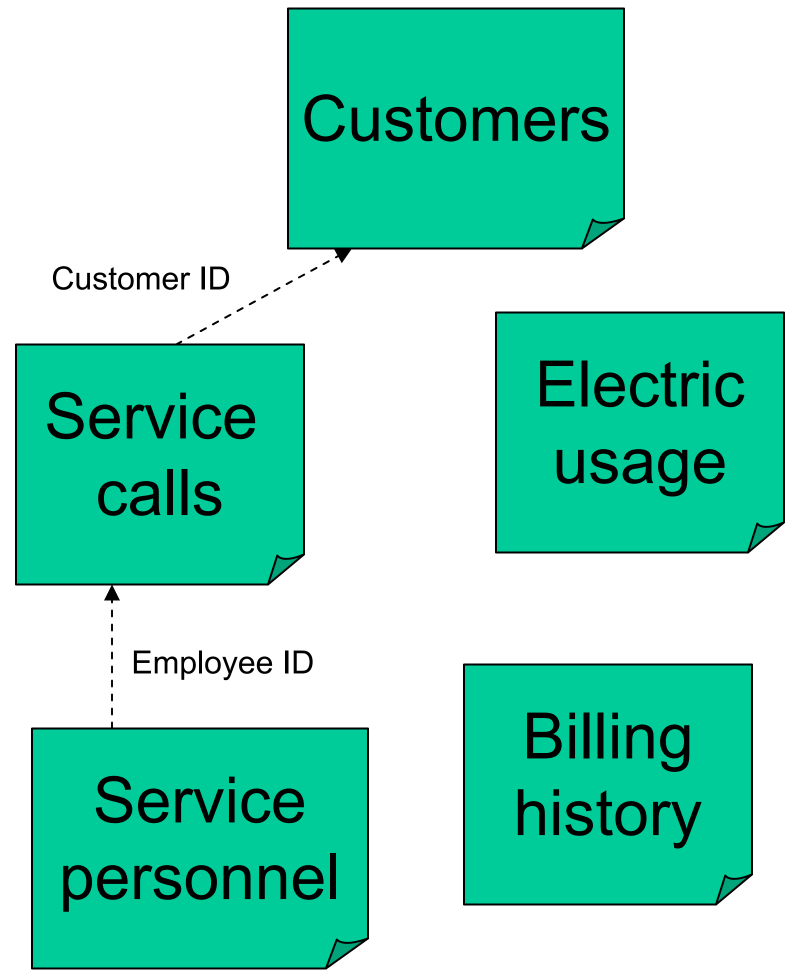


How many customers exceed 100 KwH/month?

How many service calls has Customer X had?

Which serviceman performed the most calls in December?

Relational



How many different customers has each serviceman seen?

Has Serviceman Smith ever visited Customer Jones?

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

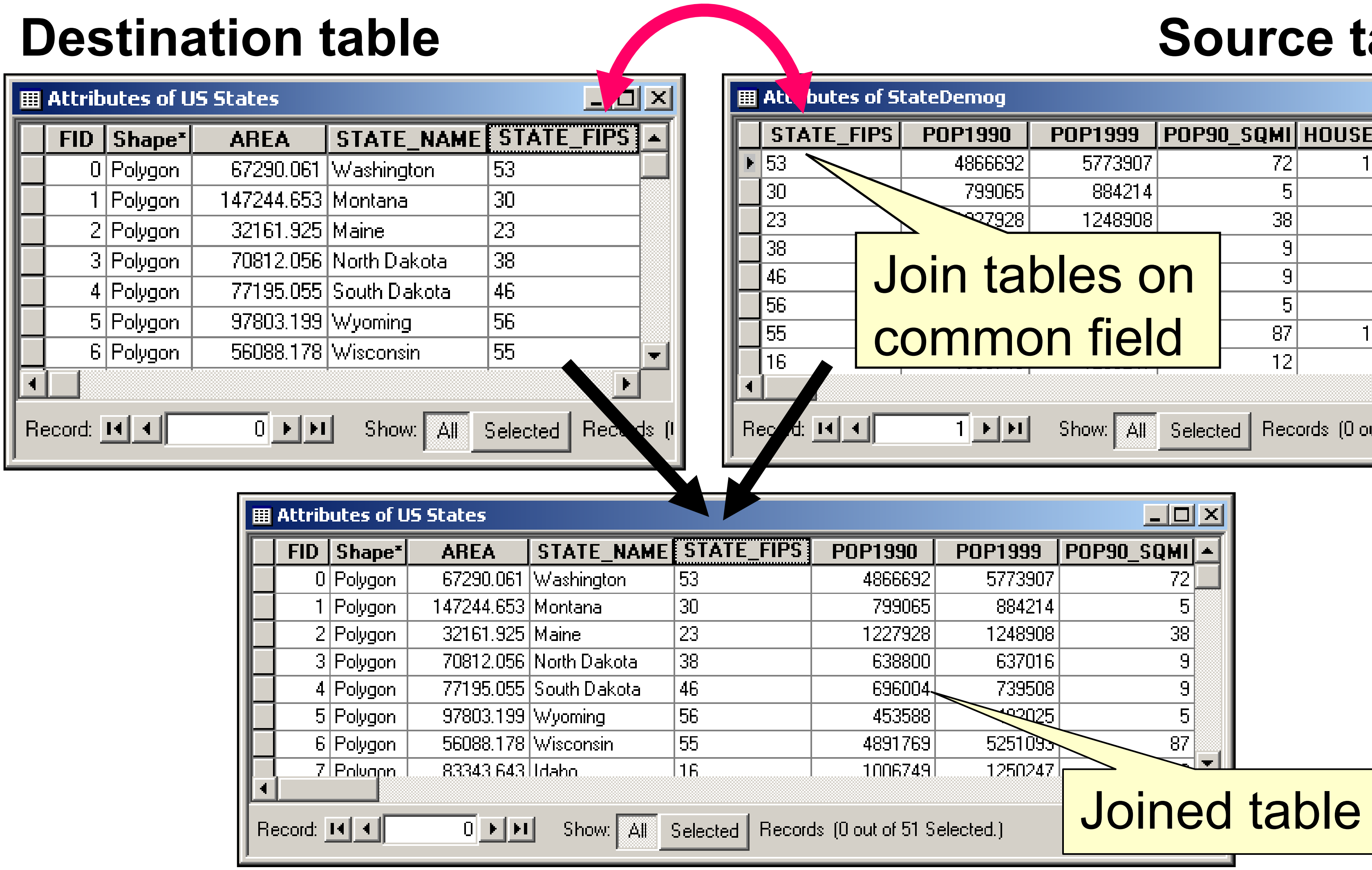
Normalization

- 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**

Joining tables



Destination table

FID	Shape*	AREA	STATE_NAME	STATE_FIPS
0	Polygon	67290.061	Washington	53
1	Polygon	147244.653	Montana	30
2	Polygon	32161.925	Maine	23
3	Polygon	70812.056	North Dakota	38
4	Polygon	77195.055	South Dakota	46
5	Polygon	97803.199	Wyoming	56
6	Polygon	56088.178	Wisconsin	55

Source table

STATE_FIPS	POP1990	POP1999	POP90_SQMI	HOUSEHOLD
53	4866692	5773907	72	1872431
30	799065	884214	5	306163
23	[illegible]	1248908	38	465312
38	[illegible]	[illegible]	9	240878
46	[illegible]	[illegible]	9	259034
56	[illegible]	[illegible]	5	168839
55	[illegible]	[illegible]	87	1822118
16	[illegible]	[illegible]	12	360723

Joined table

FID	Shape*	AREA	STATE_NAME	STATE_FIPS	POP1990	POP1999	POP90_SQMI
0	Polygon	67290.061	Washington	53	4866692	5773907	72
1	Polygon	147244.653	Montana	30	799065	884214	5
2	Polygon	32161.925	Maine	23	1227928	1248908	38
3	Polygon	70812.056	North Dakota	38	638800	637016	9
4	Polygon	77195.055	South Dakota	46	696004	739508	9
5	Polygon	97803.199	Wyoming	56	453588	[illegible]	5
6	Polygon	56088.178	Wisconsin	55	4891769	[illegible]	87
7	Polygon	83343.643	Idaho	16	1006749	1250247	[illegible]

Join facts

- Joins are **temporary relationships** between tables used by a relational DBMS
- Tables **must share a common field** (key)
- Treats the **two tables as a single table**
- Original stored data is **not affected**
- Can be removed when **no longer needed**

Relational databases

Restaurant table

Restaurant	Res-ID	Parcel_no
Jake's Pizza	20	45-98764
Momma's Pie Hut	30	64-56790
Big Burger Barn	40	62-98754

Employee table

Res-ID	Name	SSN
20	Jake Smith	134-56-7689
20	Nancy Gold	229-69-3490
20	Dan Smurt	345-34-8968
30	Karen White	776-67-4578
40	Judy Lewis	670-45-6890
40	Joshua Jones	675-56-4982

Parcels table

Parcel_no	Address	Value	Owner
45-98764	1104 Maple Ave	67,000	Roger Clark
64-56790	1900 Main St	114,510	Roger Clark
62-98754	9207 Sherry Ave	59,000	Judy Lewis

Store distinct tables

Establish relationships between them

One-to-one joins

Destination table
(always imagine on the left)

Attributes of US States

FID	Shape*	STATE_NAME	STATE_ABBR
0	Polygon	Hawaii	HI
1	Polygon	Washington	WA
2	Polygon	Montana	MT
3	Polygon	Maine	ME
4	Polygon	North Dakota	ND
5	Polygon	South Dakota	SD
6	Polygon	Wyoming	WY
7	Polygon	Wisconsin	WI

Source table
(always imagine on the right)

quakesum

STATE	Count_	Sum_DAMAGE	Su
CA	218	3705234000	
AK	106	32600000	
MT	62	4220000	
WA	67	3775000	
ID	41	1350000	
HI	63	1100000	
OR	24	760000	

When **each record** in the destination table **matches exactly one record** in the source table, we call it a **cardinality** of **one-to-one**.

Types of Cardinality

- **One-to-one**
  - States to Governors
  - Husbands to wives
- **One-to-many**
  - States to cities
  - Districts to schools

(Destination on the left)

- **Many-to-one**
  - Cities to states
  - Schools to districts
- **Many-to-many**
  - Students to classes
  - Stores to customers

In evaluating **cardinality**, always put the **destination first**.

Rule of Joining

Each record in the destination table must **match one and only one record** in the source table.

Attributes of US States

	FID	Shape*	STATE_NAME	STATE_ABBR
▸	0	Polygon	Hawaii	HI
	1	Polygon	Washington	WA
	2	Polygon	Montana	MT
	3	Polygon	Maine	ME
	4	Polygon	North Dakota	ND
	5	Polygon	South Dakota	SD
	6	Polygon	Wyoming	WY
	7	Polygon	Wisconsin	WI

quakesum

STATE	Count_	Sum_DAMAGE	Sum_DEA
CA	218	3705234000	
AK	106	32600000	
MT	62	4220000	
WA	67	3775000	
ID	41	1350000	
HI	63	1100000	
OR	24	760000	

One-to-one

Destination table

Source table

US Counties

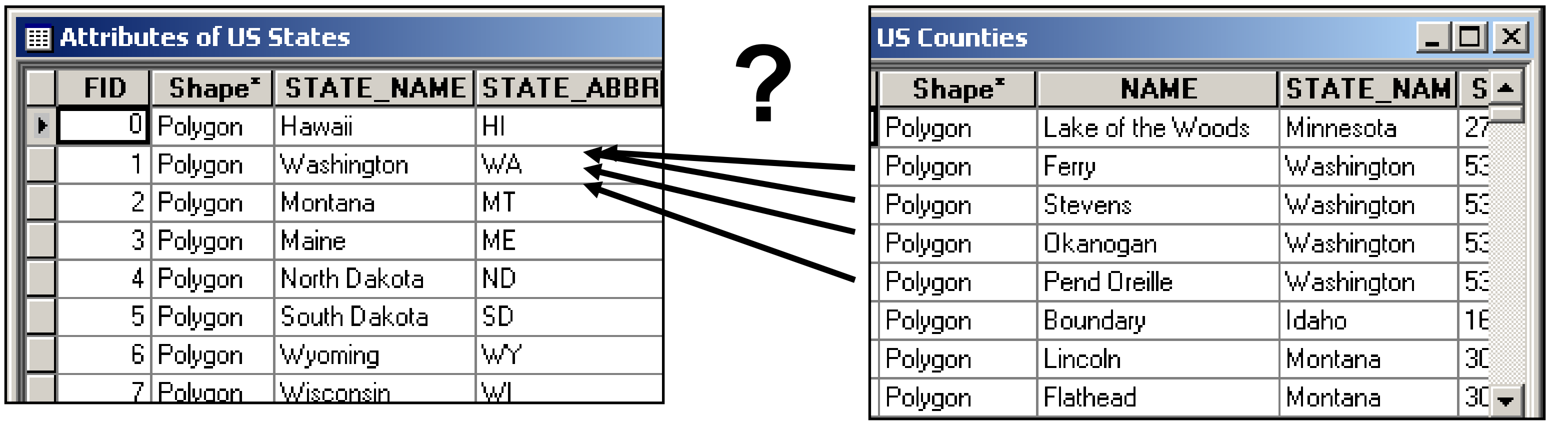
Shape*	NAME	STATE_NAM	S
Polygon	Lake of the Woods	Minnesota	27
Polygon	Ferry	Washington	53
Polygon	Stevens	Washington	53
Polygon	Okanogan	Washington	53
Polygon	Pend Oreille	Washington	53
Polygon	Boundary	Idaho	16
Polygon	Lincoln	Montana	30
Polygon	Flathead	Montana	30

Attributes of US States

	FID	Shape*	STATE_NAME	STATE_ABBR
▸	0	Polygon	Hawaii	HI
	1	Polygon	Washington	WA
	2	Polygon	Montana	MT
	3	Polygon	Maine	ME
	4	Polygon	North Dakota	ND
	5	Polygon	South Dakota	SD
	6	Polygon	Wyoming	WY
	7	Polygon	Wisconsin	WI

Many-to-one

One-to-many



Attributes of US States

FID	Shape*	STATE_NAME	STATE_ABBR
0	Polygon	Hawaii	HI
1	Polygon	Washington	WA
2	Polygon	Montana	MT
3	Polygon	Maine	ME
4	Polygon	North Dakota	ND
5	Polygon	South Dakota	SD
6	Polygon	Wyoming	WY
7	Polygon	Wisconsin	WI

US Counties

Shape*	NAME	STATE_NAM	S
Polygon	Lake of the Woods	Minnesota	27
Polygon	Ferry	Washington	53
Polygon	Stevens	Washington	53
Polygon	Okanogan	Washington	53
Polygon	Pend Oreille	Washington	53
Polygon	Boundary	Idaho	16
Polygon	Lincoln	Montana	30
Polygon	Flathead	Montana	30

Destination table

Source table

Violates the Rule of Joining

Record to join to destination is **ambiguous**

Must use a **relate** instead

Relates

- **Similar to a join** except that
  - The tables **remain separate**
  - Items **selected in one table** may be **highlighted in the related table**

Related tables

States: Select the New England States

Attributes of states

OBJECTID	Shape *	AREA	STATE_NAME	STATE_FIPS	SUB_REGION	STATE_A
8	Polygon	56088.178	Wisconsin	55	E N Cen	WI
9	Polygon	83343.643	Idaho	16	Mtn	ID
10	Polygon	9603.272	Vermont	50	N Eng	VT
11	Polygon	84520.49	Minnesota	27	W N Cen	MN
12	Polygon	97073.594	Oregon	41	Pacific	OR
13	Polygon	9259.527	New Hampshire	33	N Eng	NH
14	Polygon	56257.965	Iowa	19	W N Cen	IA
15	Polygon	8172.561	Massachusetts	25	N Eng	MA
16	Polygon	77330.258	Nebraska	31	W N Cen	NE

Record: 1 Show: All Selected Records (6 out of 51 Selected)

- Find & Replace...
- Select By Attributes...
- Select All
- Clear Selection
- Switch Selection
- Add Field...
- Related Tables ▸
  - States-To-Districts : cd108

Congress Reps of New England States

Attributes of cd108

OBJECTID *	Shape *	NAME	PARTY	DISTRICTID	STFIPS	STATE_ABBR
413	Polygon	Carolyn McCarthy	Democrat	3604	36	NY
414	Polygon	Steve J. Israel	Democrat	3602	36	NY
415	Polygon	Peter T. King	Republican	3603	36	NY
416	Polygon	James McGovern	Democrat	2503	25	MA
417	Polygon	Rob Simmons	Republican	0902	09	CT
418	Polygon	Patrick J. Kennedy	Democrat	4401	44	RI
419	Polygon	James R. Langevin	Democrat	4402	44	RI
420	Polygon	Rosa L. DeLauro	Democrat	0903	09	CT

Record: 1 Show: All Selected Records (22 out of 437 Selected) Options

Summarizing tables

- Calculate statistics for **groups of features** in a table
- **Groups by unique values** in the one field
- User **chooses statistics** to calculate for other fields
- **Produces another table as output** with groups and stats

AREA	STATE_NAME	STATE_FIPS	SUB_REGION	STATE_ABBR	POP1990
56088.178	Wisconsin	55	E N Cen	WI	4891769
83343.643	Idaho	16	Mtn	ID	1006749
9603.272	Vermont	50	N Eng	VT	562758
84520.49	Minnesota	27	W N Cen	MN	4375099
97073.594	Oregon	41	Pacific	OR	2842321
9259.527	New Hampshire	33	N Eng	NH	1109252
56257.965	Iowa	19	W N Cen	IA	2776755
8172.561	Massachusetts	25	N Eng	MA	6016425
77330.258	Nebraska	31	W N Cen	NE	1578385

How many people live **in each subregion**?

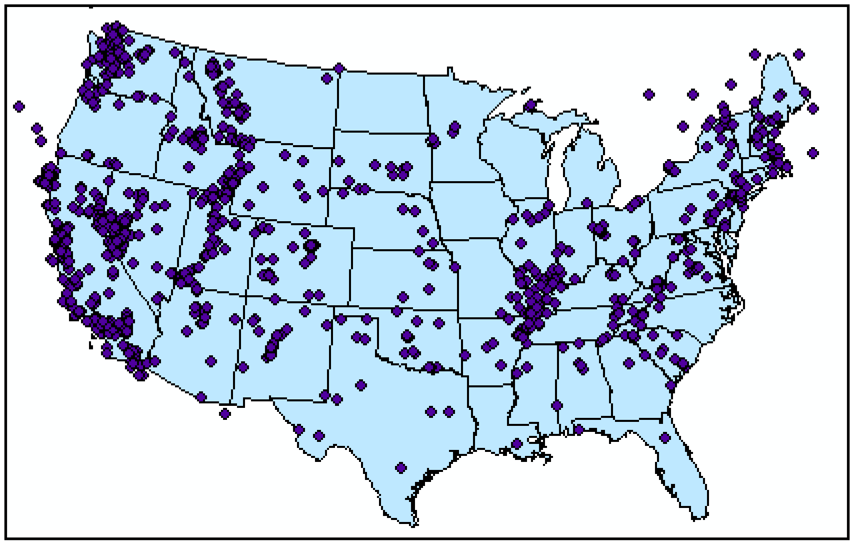
What is the **total area** of **each subregion**?

Summarizing tables

Attributes of quakehis

STATE	DEPTH	DEATHS	DAMAGE	MAG	MMI	LOCATION
CA	20	3000	52400000	7.8	11	Near San Francisco, California
CA	0	300	0	0	9	Southern California
AK	33	125	31100000	9.23	10	Prince William Sound, Alaska
PR	0	116	400000	7.5	9	Northwestern Mona Passage
CA	16	115	4000000	6.3	8	Southeast of Long Beach, near Newport Beach, California
HI	0	77	0	7.9	10	Near south coast of Hawaii

Historic major earthquakes

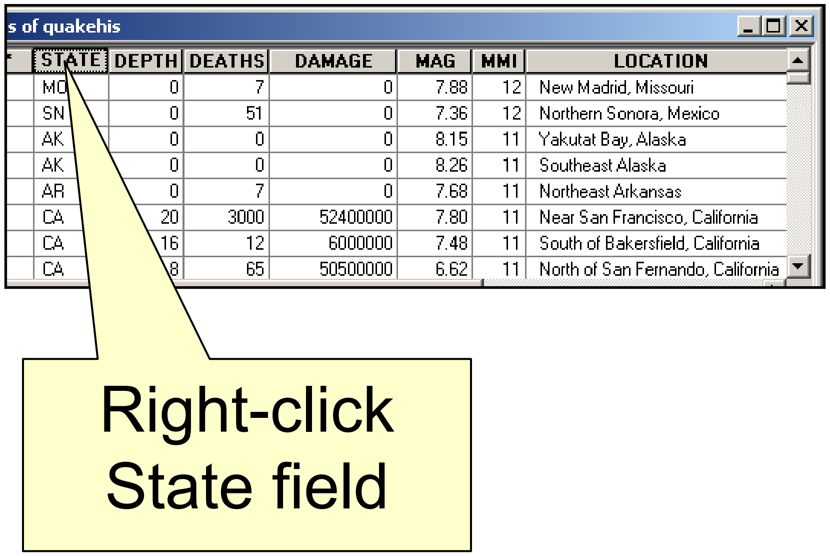


How many earthquakes in each state?

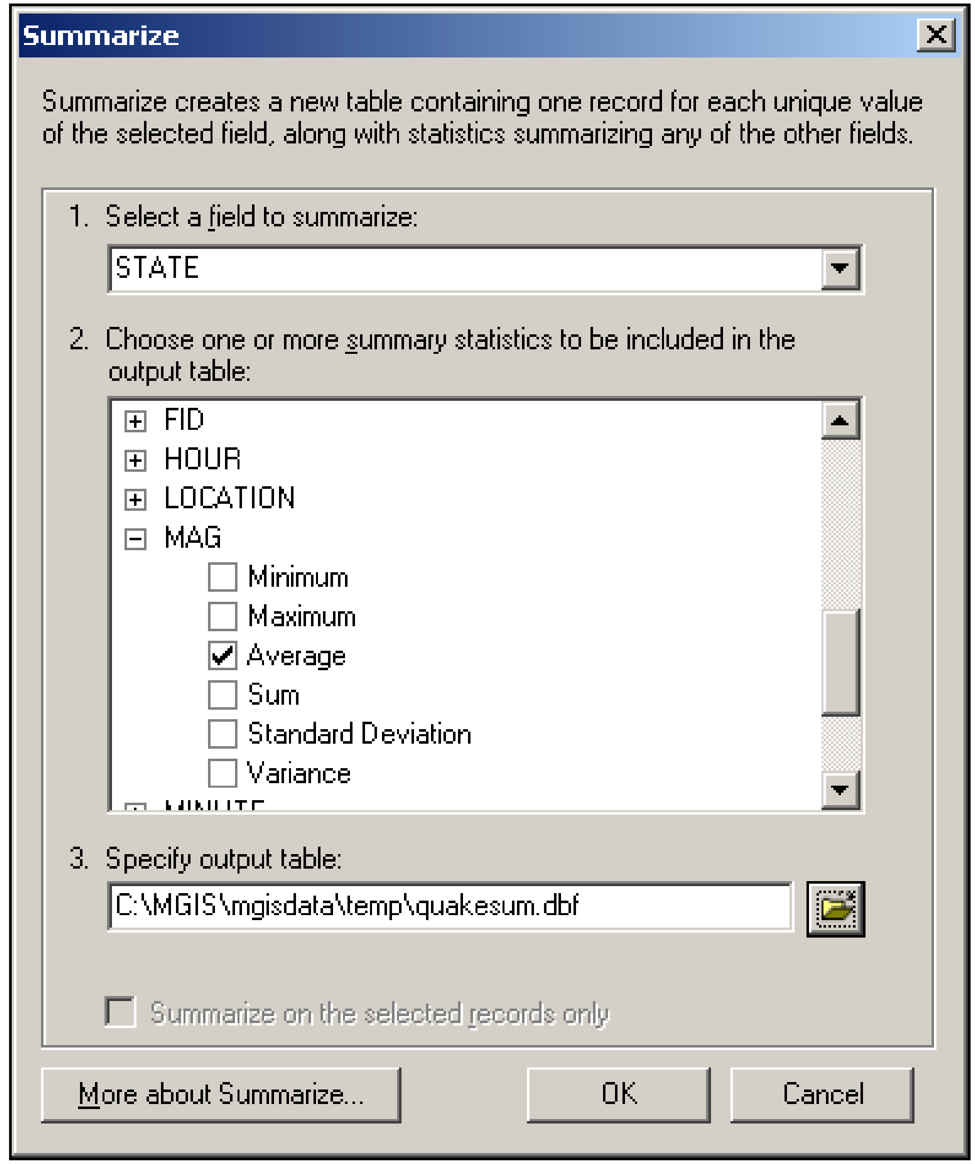
Total deaths and damage in each state?

Average magnitude in each state?

How to summarize



STATE	DEPTH	DEATHS	DAMAGE	MAG	MMI	LOCATION
MO	0	7	0	7.88	12	New Madrid, Missouri
SN	0	51	0	7.36	12	Northern Sonora, Mexico
AK	0	0	0	8.15	11	Yakutat Bay, Alaska
AK	0	0	0	8.26	11	Southeast Alaska
AR	0	7	0	7.68	11	Northeast Arkansas
CA	20	3000	52400000	7.80	11	Near San Francisco, California
CA	16	12	6000000	7.48	11	South of Bakersfield, California
CA	8	65	50500000	6.62	11	North of San Fernando, California



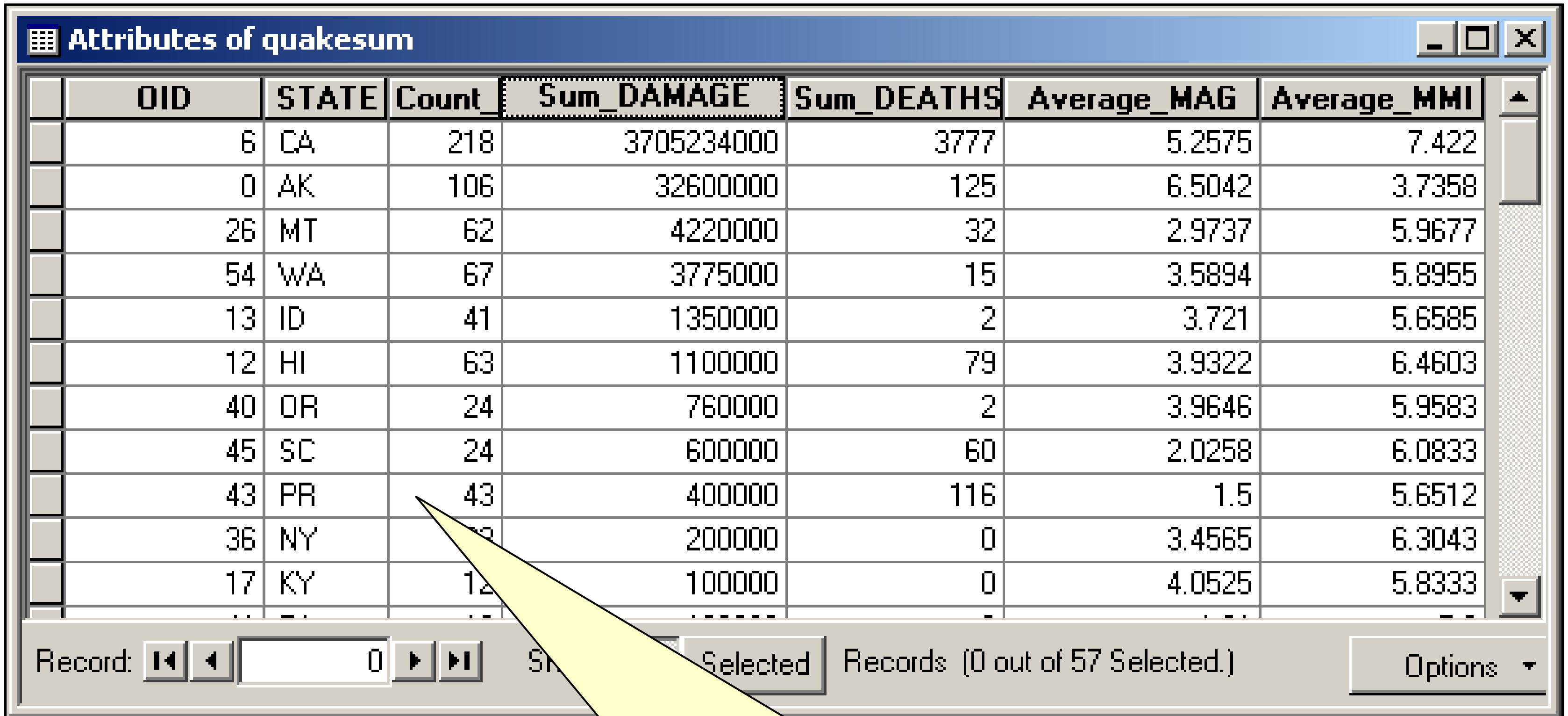
Sum Deaths

Sum Damage

Average Mag

Average MMI

Summarize Output Table



Attributes of quakesum

OID	STATE	Count_	Sum_DAMAGE	Sum_DEATHS	Average_MAG	Average_MMI
6	CA	218	3705234000	3777	5.2575	7.422
0	AK	106	32600000	125	6.5042	3.7358
26	MT	62	4220000	32	2.9737	5.9677
54	WA	67	3775000	15	3.5894	5.8955
13	ID	41	1350000	2	3.721	5.6585
12	HI	63	1100000	79	3.9322	6.4603
40	OR	24	760000	2	3.9646	5.9583
45	SC	24	600000	60	2.0258	6.0833
43	PR	43	400000	116	1.5	5.6512
36	NY	[illegible]	200000	0	3.4565	6.3043
17	KY	1[illegible]	100000	0	4.0525	5.8333

Record: 0 Selected Records (0 out of 57 Selected.) Options

Count field always generated automatically

Create map

- Could we now **create a map** of deaths by state?
  - **No**, there are **no features** (yet).

Standalone table

Attributes of quakesum

OID	STATE	Count_	Sum_DAMAGE	Sum_DEATHS	Average_MAG	Average_MMI
6	CA	218	3705234000	3777	5.2575	7.422
0	AK	106	32600000	125	6.5042	3.7358
26	MT	62	4220000	32	2.9737	5.9677
54	WA	67	3775000	15	3.5894	5.8955
13	ID	41	1350000	2	3.721	5.6585
12	HI	63	1100000	79	3.9322	6.4603
40	OR	24	760000	2	3.9646	5.9583
45	SC	24	600000	60	2.0258	6.0833
43	PR	43	400000	116	1.5	5.6512
36	NY	23	200000	0	3.4565	6.3043
17	KY	12	100000	0	4.0525	5.8333

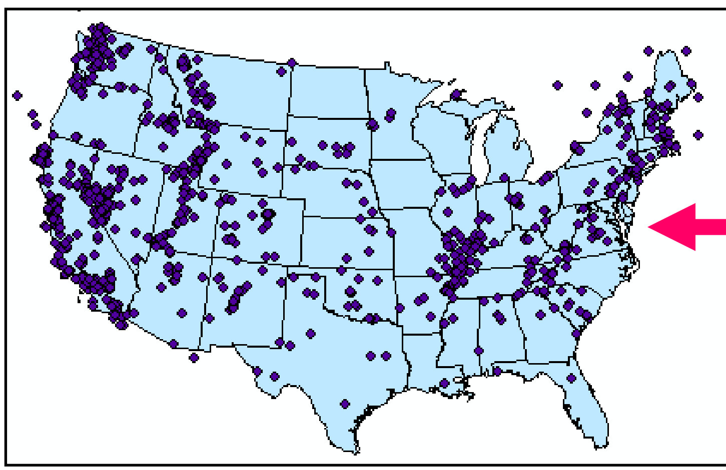
Record: 0 Show: All Selected Records (0 out of 57 Selected.) Options

Joining the table

Summarize output table

Attributes of quakesum

ID	STATE	Count_	Sum_DAMAGE	Sum_DEATHS	A
6	CA	218	3705234000	3777	
0	AK	106	32600000	125	
26	MT	62	4220000	32	
54	WA	67	3775000	15	
13	ID	41	1350000	2	
12	HI	63	1100000	79	
40	OR	24	760000	2	

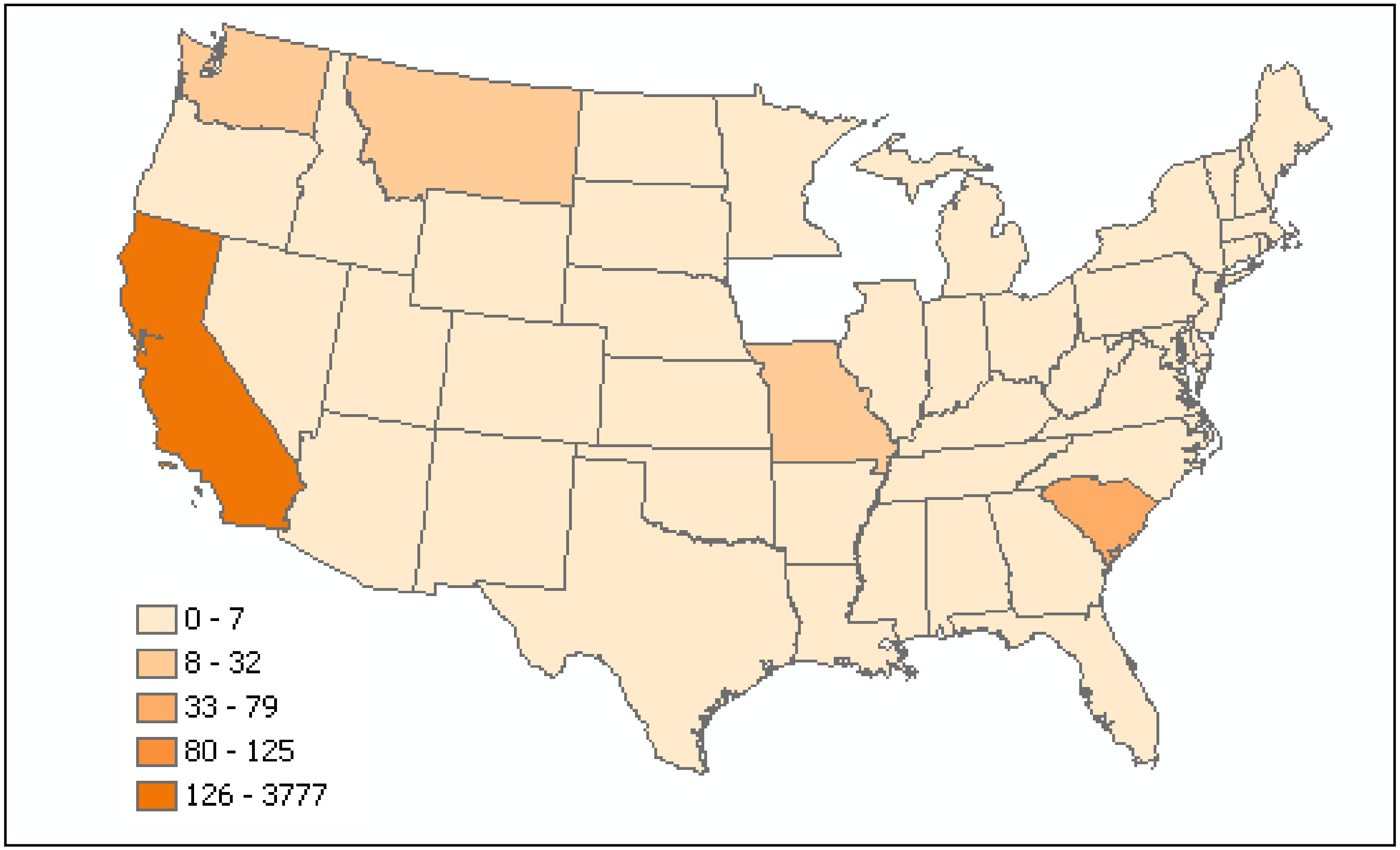


States layer attributes

Attributes of US States

FID	Shape*	STATE_NAME	STATE	
0	Polygon	Hawaii	HI	1108229
1	Polygon	Washington	WA	4866692
2	Polygon	Montana	MT	799065
3	Polygon	Maine	ME	1227928
4	Polygon	North Dakota	ND	638800
5	Polygon	South Dakota	SD	696004
6	Polygon	Wyoming	WY	453588
7	Polygon	Wisconsin	WI	4891769

US Earthquake Deaths by State



Join summarize output to states layer to **create map** of deaths

Fields

- Fields have **specific types available**
- Must be **defined before use**
- Once defined, **cannot be changed**
- **Naming rules**
  - **No more** than **13 characters**
  - Use **only letters and numbers**
  - Must **start with a letter**
- How is information **actually stored** in fields …

Maps as Numbers

- GIS requires that both data and maps be **represented as numbers**.
- The GIS places data into the computer's memory in a **physical data structure** (i.e. files and directories).
- Files can be written in **binary** or as **ASCII text**.
- Binary is **faster to read and smaller**, ASCII can be **read by humans and edited** but uses more space.

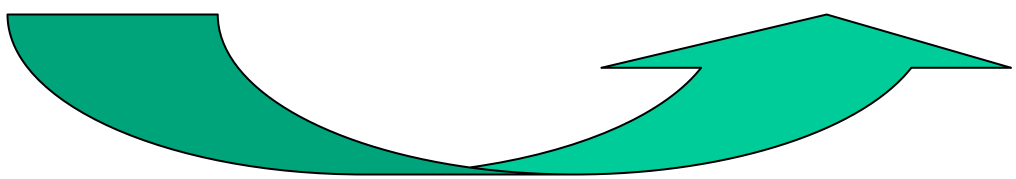
Binary Notation

•Everything is represented as 0s and 1s in a computer. These two-state forms correspond to yes/no, on/off, open/closed

	Binary		Decimal
1 digit	0, 1	1 bit	0,1,2,…9
2 digits	00, 01	2 bits	00, 01,…
	10, 11		97, 99
3 digits	000, 001	3 bits	000, 001,
	010, 011		002, 003,
	100, 101		…
	110, 111		998, 999

One to one correspondence

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	?



Binary Notation

Decimal:

$72{,}479 = 70{,}000 = 7\times10^4$

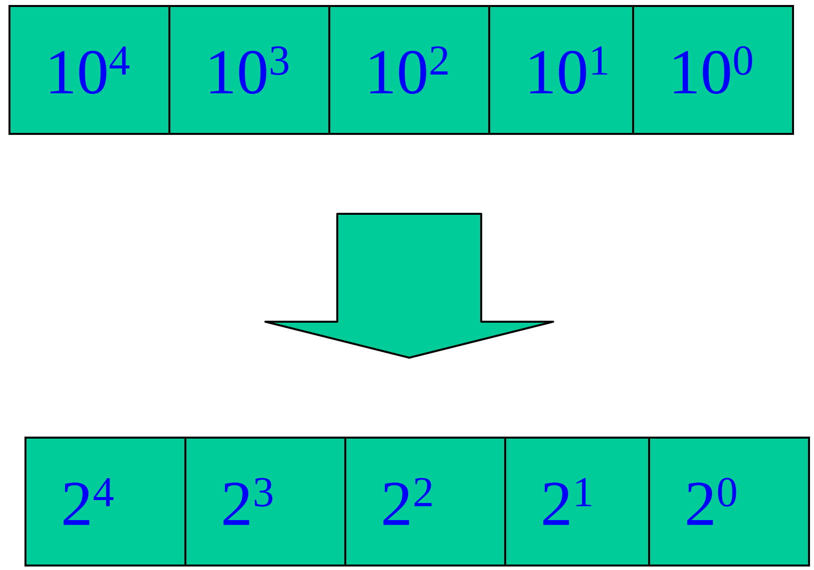
$2{,}000 = 2\times10^3$

$400 = 4\times10^2$

$70 = 7\times10^1$

$9 = 9\times10^0$

Binary:



1 0 1 0 0

$1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 0\times 2^0$

$= 16 + 0 + 4 + 0 + 0$

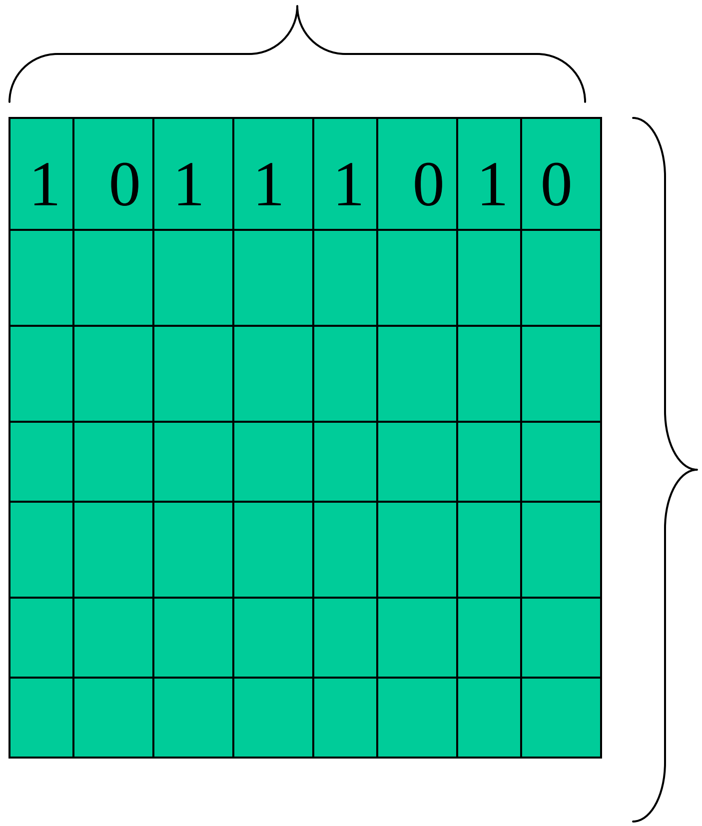
$= 20$

Note: In binary

$$\begin{array}{r} 1010 \\ +\ \ 110 \\ \hline 10000 \end{array}$$

Bits and Bytes

8 bits = 1 byte



1 bit = 1 binary digit
1 byte = 8 bits

1024 bytes = 1 Kb
1024 Kb = 1 Mb
1024 Mb = 1 Gb
1024 Gb = 1 Tb
1024 Tb = 1 Pb

ASCII Encoding

- If computers store everything using 0s and 1s, then how are **characters** represented?
- The **ASCII** (American Standard Code for Information Interchange) code assigns the numbers 0 through 127 to 128 characters, including upper and lower case alphabets plus various special characters, such as white space etc.
- e.g. decimal 85 is assigned to represent upper case U. In binary, 01010101 = 85. Thus the computer represents U using 01010101.
- Files which contain information encoded in ASCII are **easily transferred** and processed by different computers and programs. These are called “ASCII” or “text” files.

ASCII storage

- American Standard Code for Information Interchange (ASCII)
- Stores letters, characters, and symbols as **single 8-bit binary codes**

CAT = {67,65,84} decimal = 010000110100000101010100

↓ ↓ ↓

cat = {99,97,116} decimal = 011000110110000101110100

148 = {49,52,56} decimal = 001100010011010000111000

Storing data

- **Text data** always stored in **ASCII** format
- **Numeric data** may be stored in ASCII **or binary** format
- Binary is **generally more efficient**

ASCII stores **three letter codes of 1 byte each** = 3 bytes
106 = {49,48,54} decimal = 001100010011000000110110

Binary stores 106 as **a single 1-byte binary number**
106 = 01101010

Byte storage limits

- A **single byte** can store a value from **0 to 2^8-1**
- **Larger numbers** require **more bytes**
  - 1-byte $2^8-1 = 255$
  - 2-bytes $2^{16}-1 = 65,535$
  - 3-bytes $2^{24}-1 = 16,777,215$
  - 4-bytes $2^{32}-1 = 4,294,967,295$

In base 2:
00000000 = 0
11111111 = 255
$2^8 = 256$

- **Signed numbers** require a **bit to store positive or negative**, so storage limits are **smaller**
  - 2 bytes $2^{15}-1 =$ -32,767 to +32,767
  - 4 bytes $2^{31}-1 =$ -2,147,483,647 to +2,147,483,647

Integer vs. float storage

Scientific notation 3.2957239 x 10^4

- Binary stores **whole numbers** (integers)
- To store **decimal values**, the computer stores a form of scientific notation with a mantissa and an exponent
  - 3.2957239e04 = 32957.239
  - -3.2957239e04 = -32957.239
  - 3.2957239e-04 = 0.00032957239

Float precision

- **Large numbers** start to **lose precision** because the **number of significant digits in the mantissa is limited**.
  - 3.2957239e12 = 3295723900000
- A **double-precision** floating point **allots more storage to the mantissa** value
  - 3.295723956249723e12 = 3295723956249.723

Database storage

- Database fields typically are **defined by**:
  - **ASCII vs. binary** type storage
  - **Bytes of storage** allocated
  - **Integer vs. floating point**
- Definition **limits the values** that can be stored
  - Important to **match type to storage requirements**
  - Try to **minimize storage space** while making sure all **potential values will fit in the field**

Text (ASCII) field with **10 bytes** "Mississipp"

Binary **2-byte signed** integer: -32,767 to +32,767

Single-precision floating point x.xxxxxxxeyy

About ArcGIS

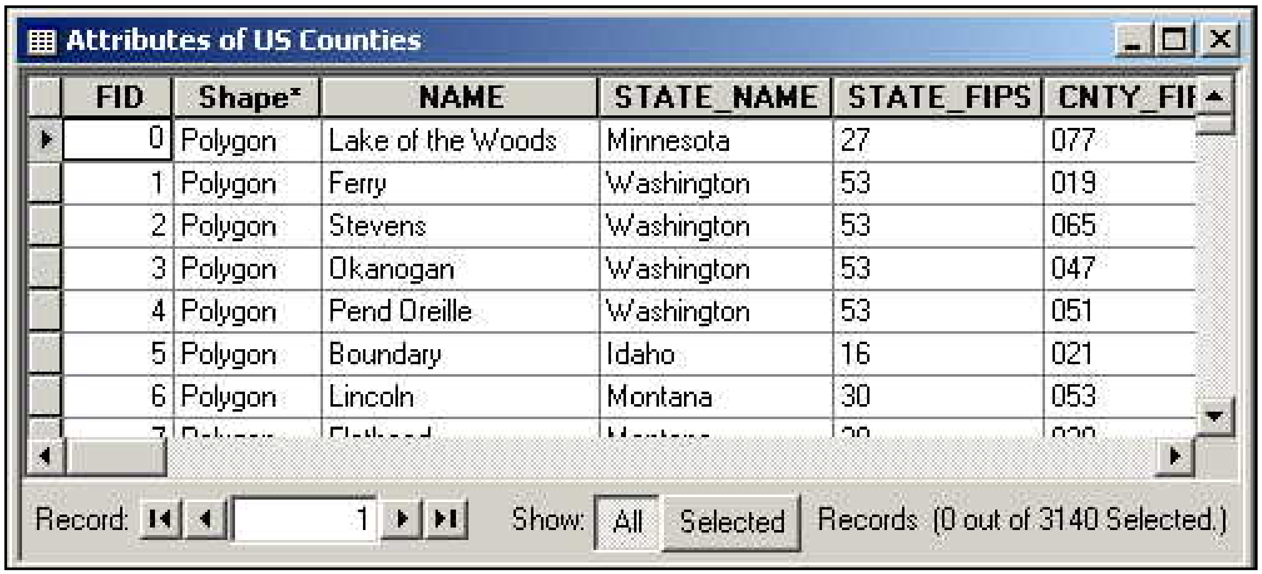
Chapter 4.

Attribute Data

Working with tables in ArcGIS

Tables in ArcGIS

- Tables contain **attribute data**
- Many formats, one interface

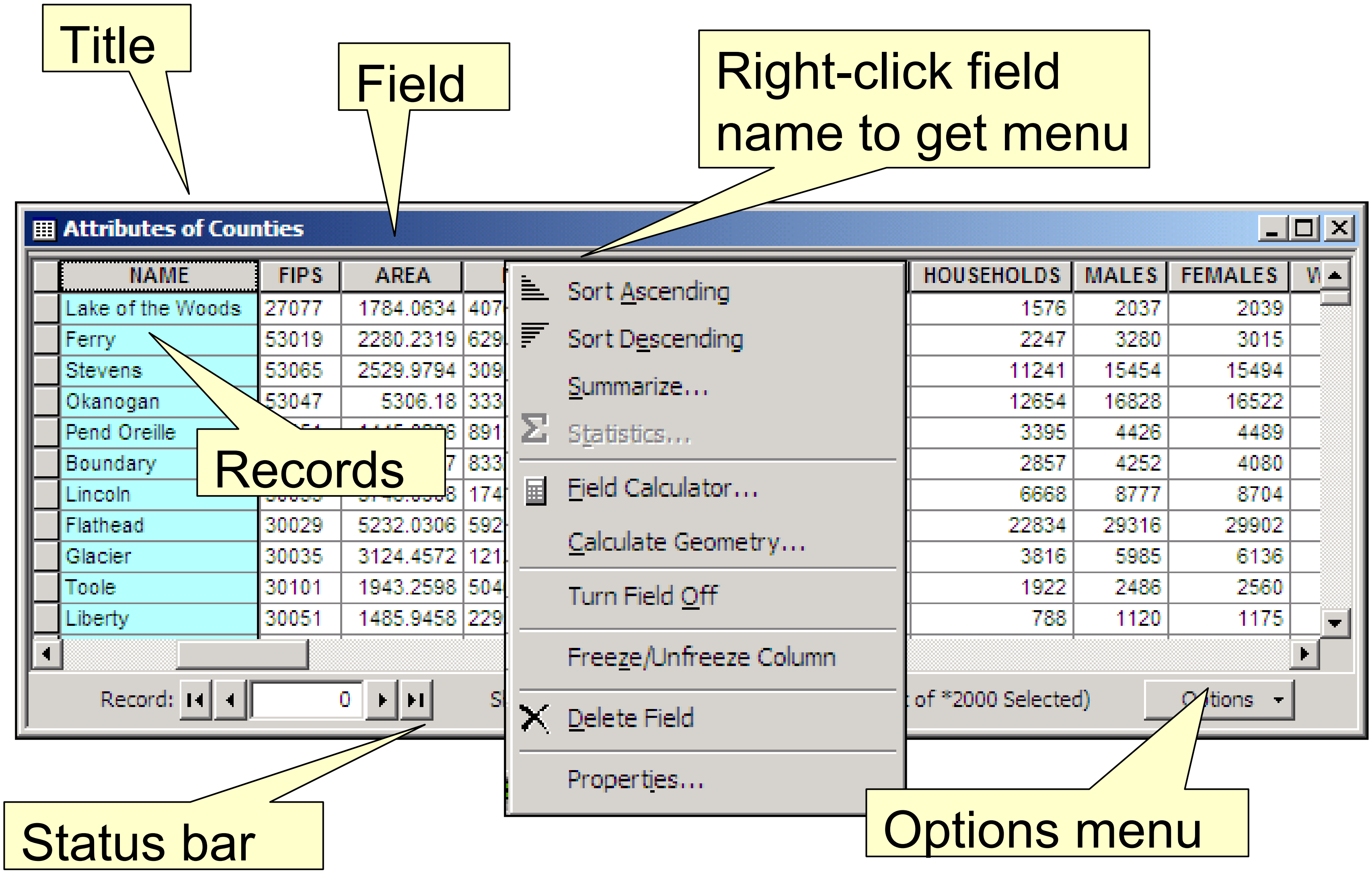


FID	Shape*	NAME	STATE_NAME	STATE_FIPS	CNTY_FIF
0	Polygon	Lake of the Woods	Minnesota	27	077
1	Polygon	Ferry	Washington	53	019
2	Polygon	Stevens	Washington	53	065
3	Polygon	Okanogan	Washington	53	047
4	Polygon	Pend Oreille	Washington	53	051
5	Polygon	Boundary	Idaho	16	021
6	Polygon	Lincoln	Montana	30	053

Sources of tables

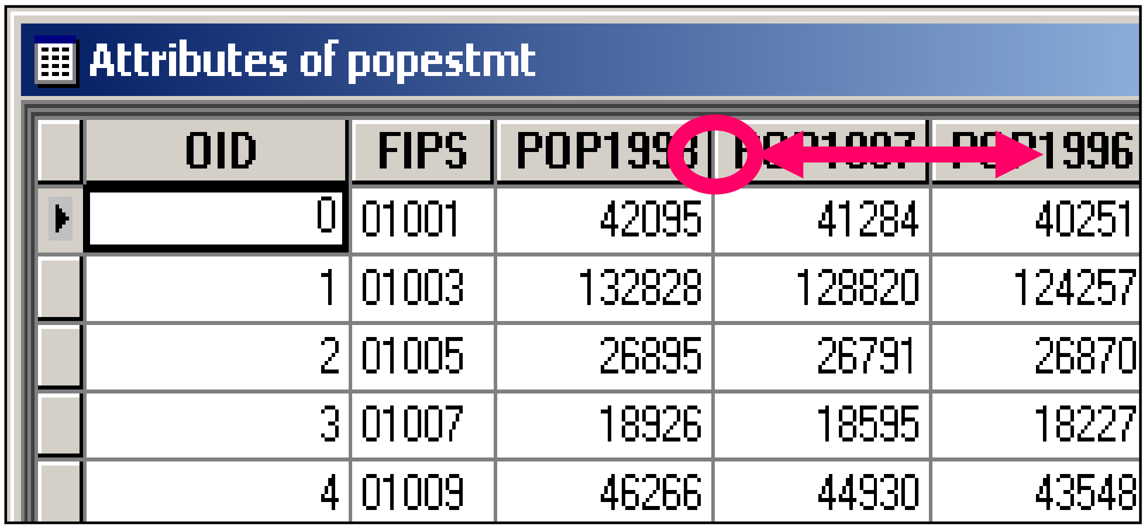
- Dbase files
- INFO files
- ASCII Text files (tab or comma delimited)
- Records from SQL database systems
- Excel worksheets

ArcMap table interface



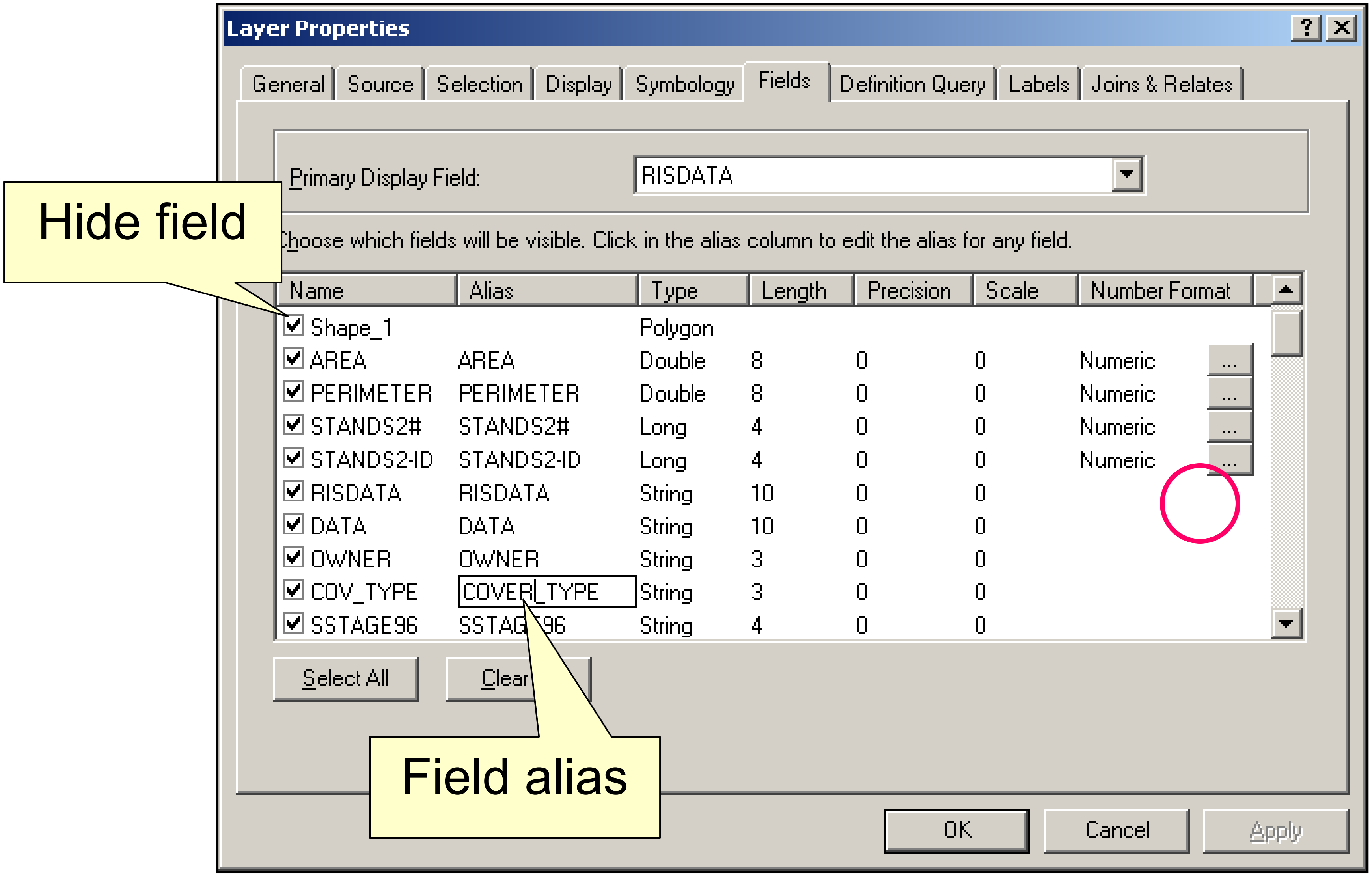
Adjusting field width

- **Temporary**, does not affect stored file

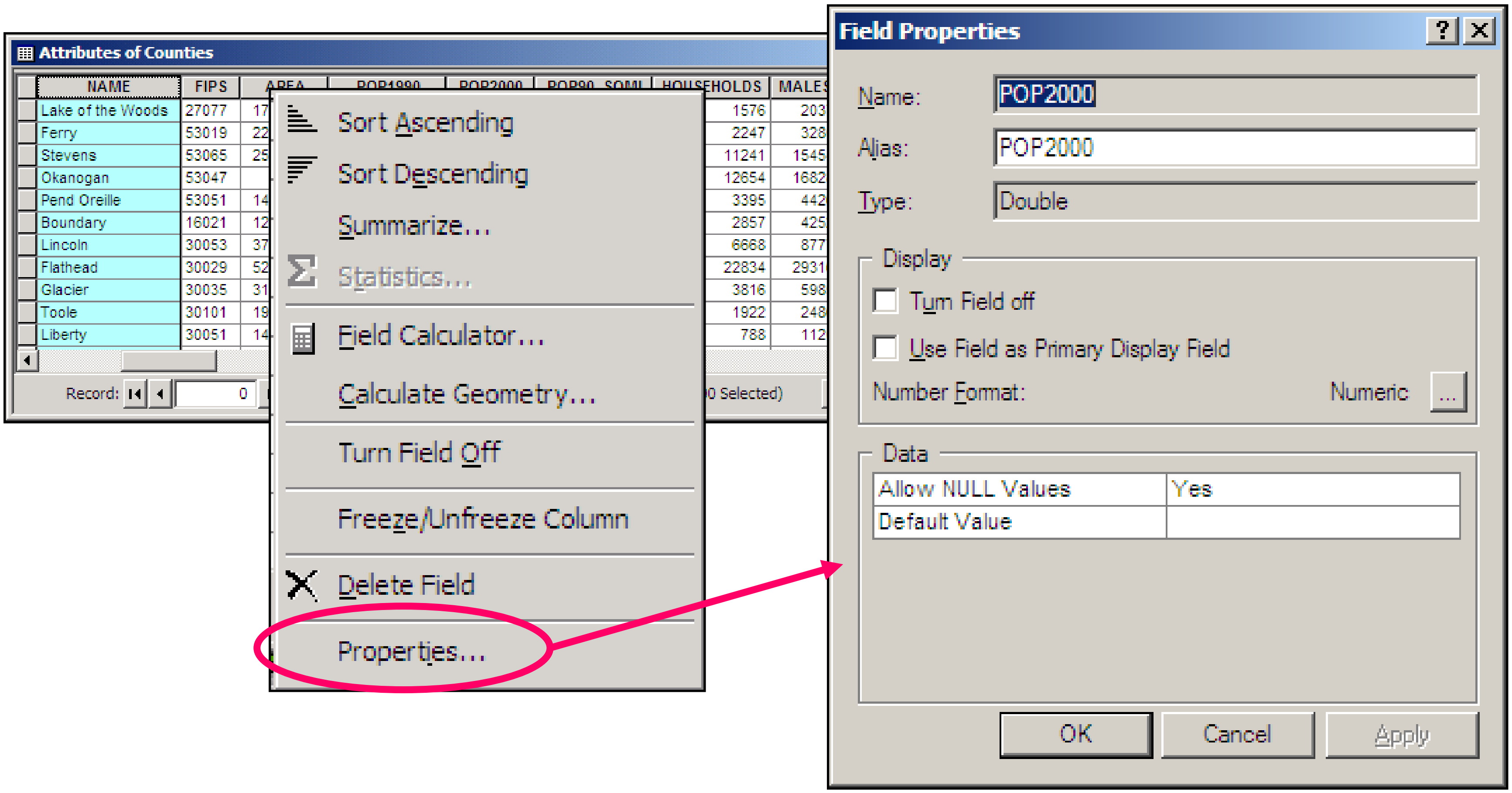


Hover over field break to get double arrow, then drag

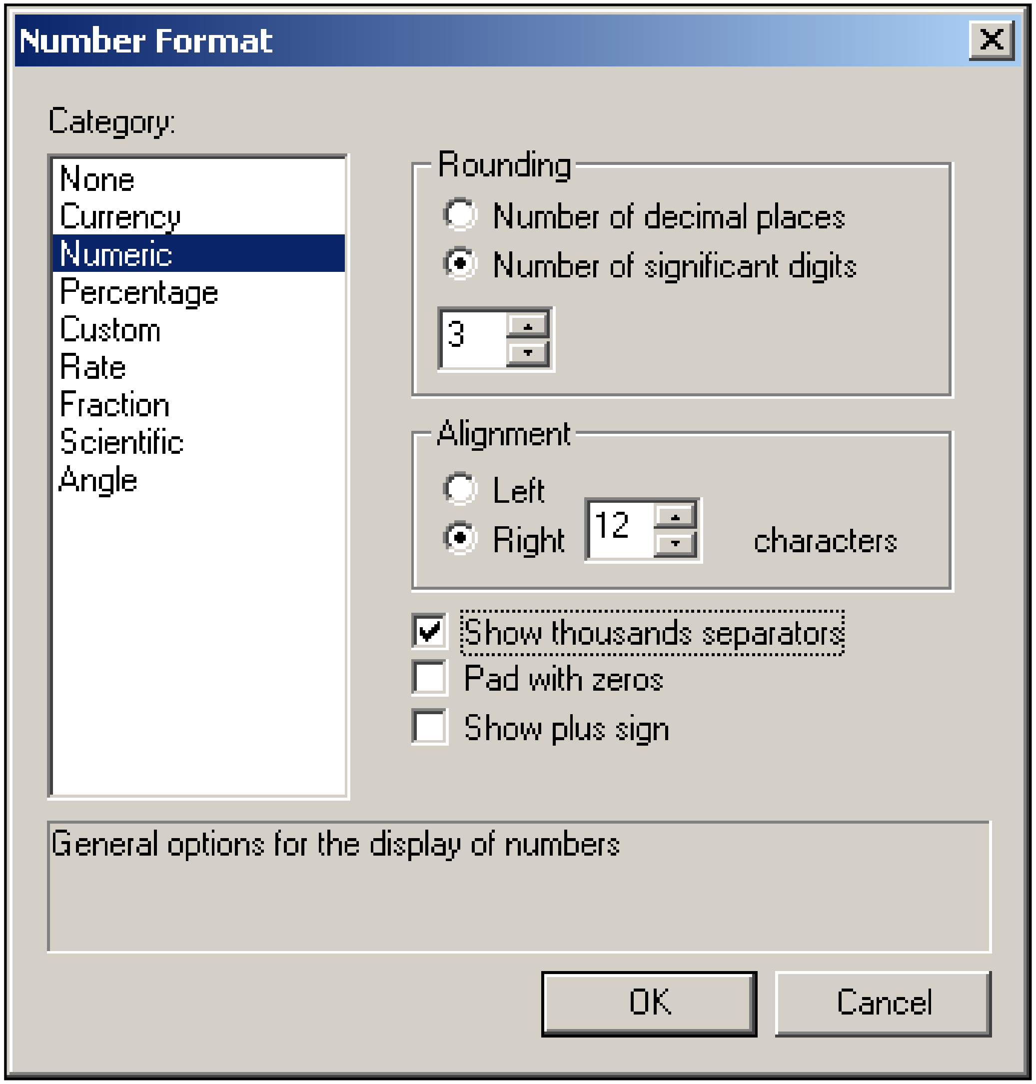
Field properties tab



Shortcut to field properties



Formatting field display



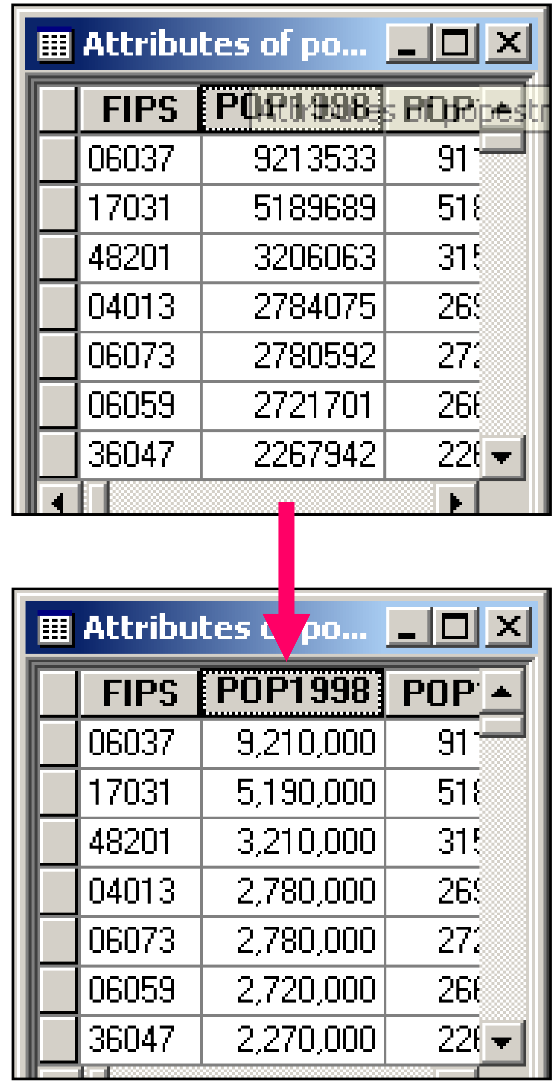
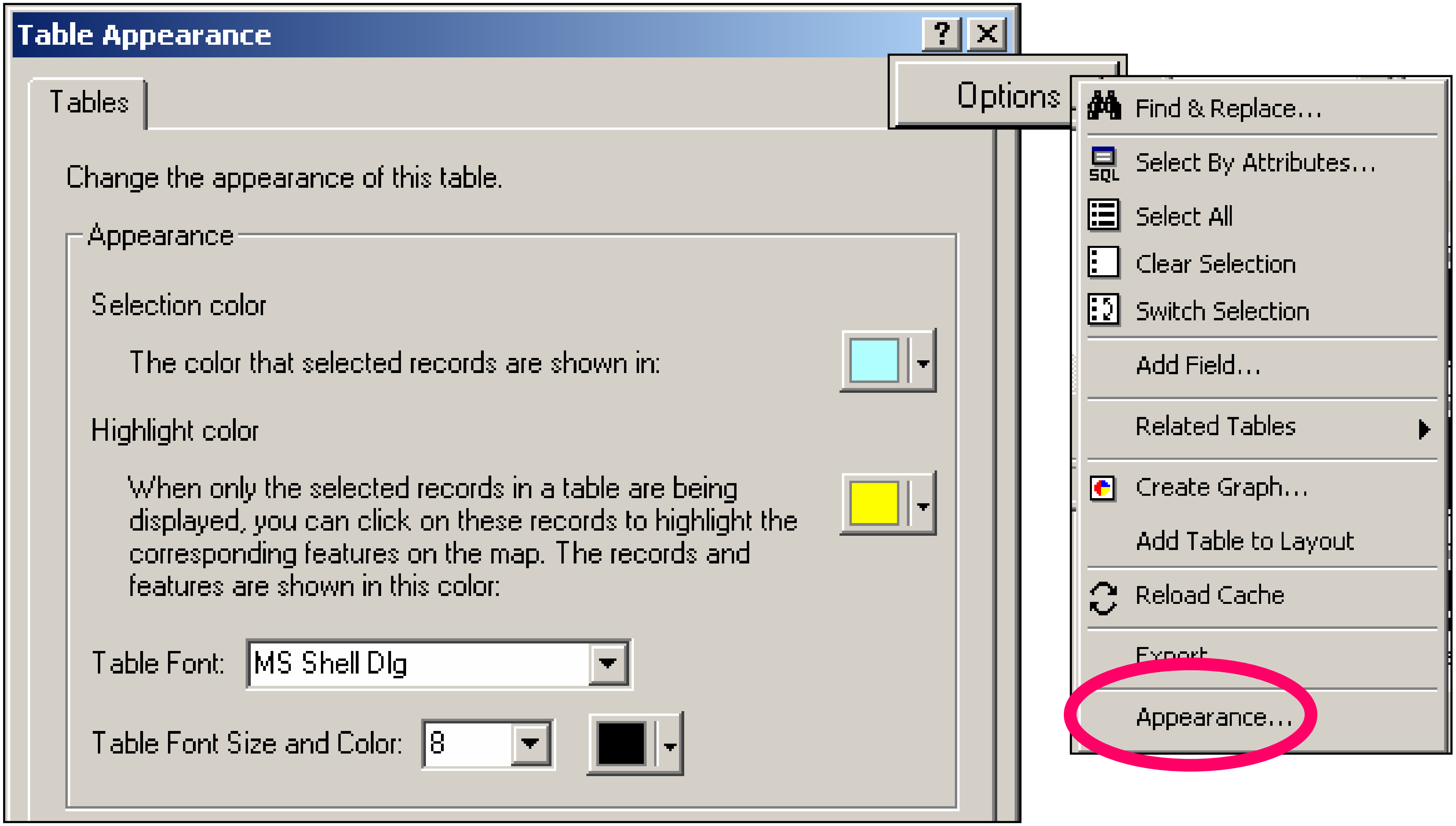
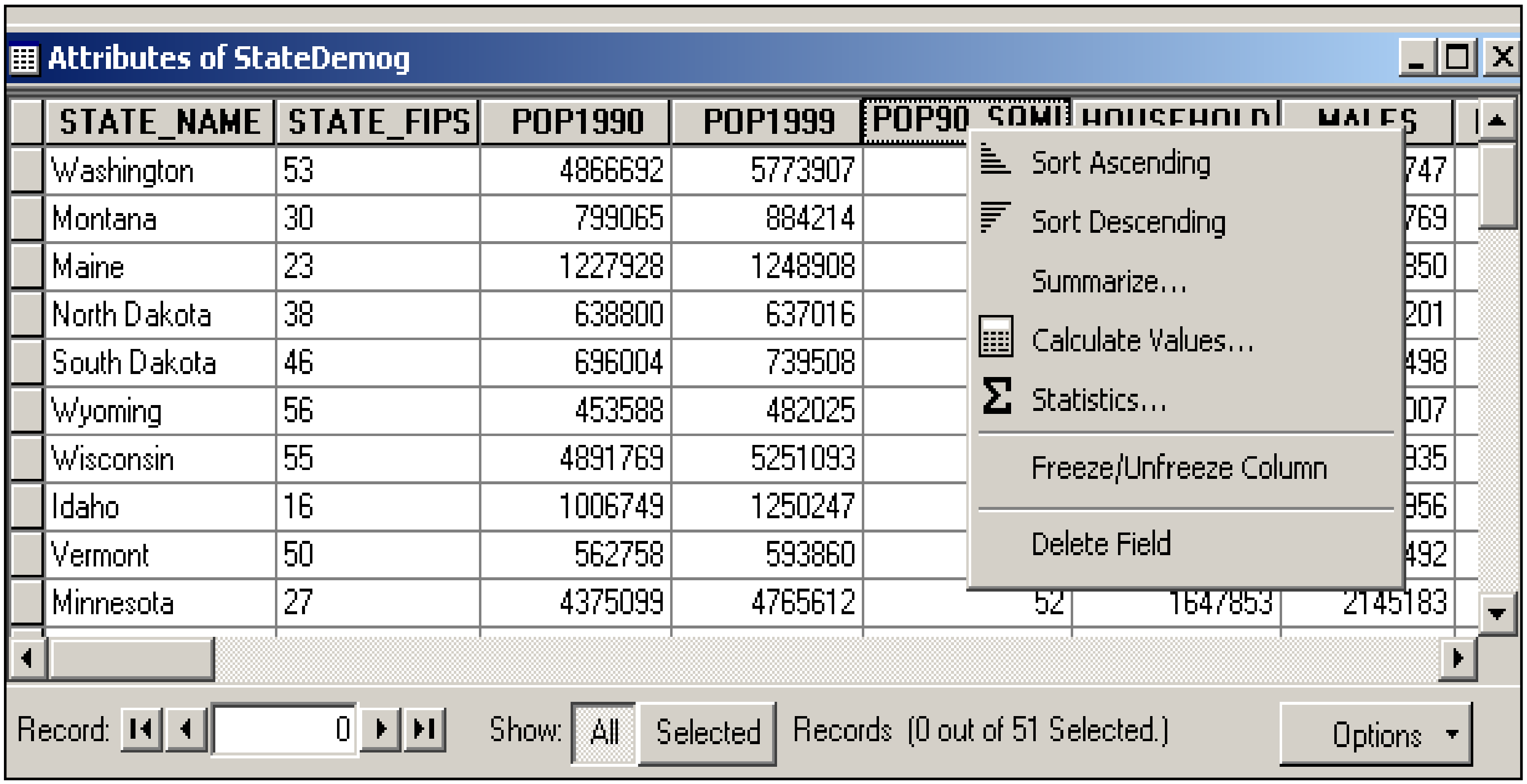


Table appearance



Sorting tables

- Has **no effect** on original data



ArcGIS field data types

Geodatabases and shapefiles		
Short	Integers stored as signed 2-byte binary numbers (value range from -32,000 to +32,000)	255 1201
Long	Integers stored as signed 4-byte binary numbers (value range from -2 billion to +2 billion)	156000
Float	Floating point values with 8 significant digits in the mantissa	1.2893851e12
Double	Double-precision floating point values with 16 significant digits in the mantissa	1.111111111111111e13
Text	Alphanumeric strings	'Maple St'
Date	Date format	07/12/92
BLOB	Binary large object; any complex binary data including images, documents, etc.	

Field characteristics

- **Length**
  - The **total characters** a text field can store
- **Precision**
  - The **total width of digits** a numeric field can store
- **Scale**
  - The number of **decimal places**

Length = 10
Maple St.
Maple Stre

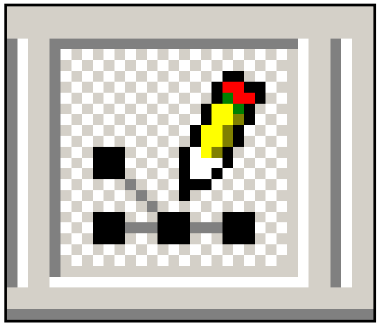
156
1985.128
-1922.5600

0.001
0.00001

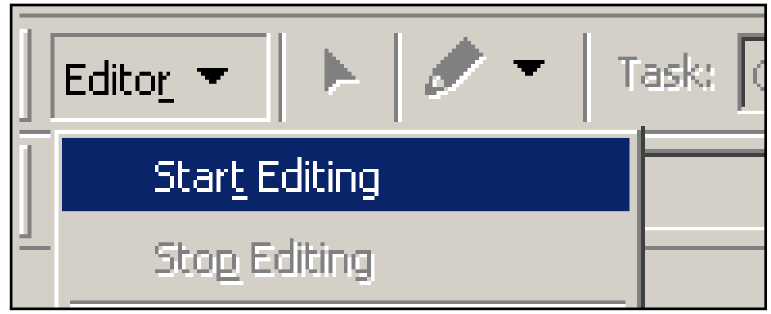
Editing and calculating fields

Editing fields

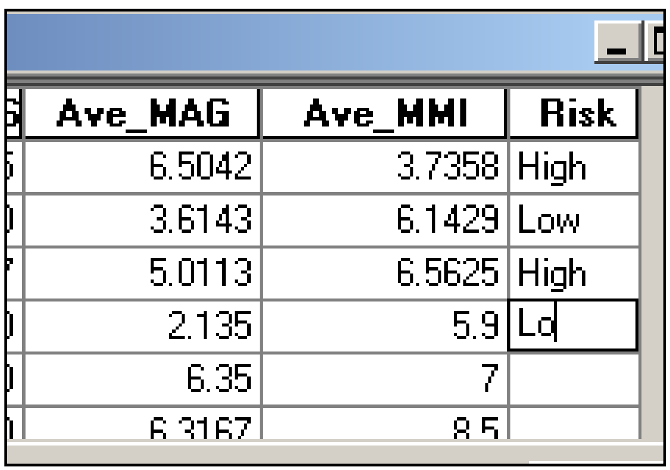
Open Editor toolbar



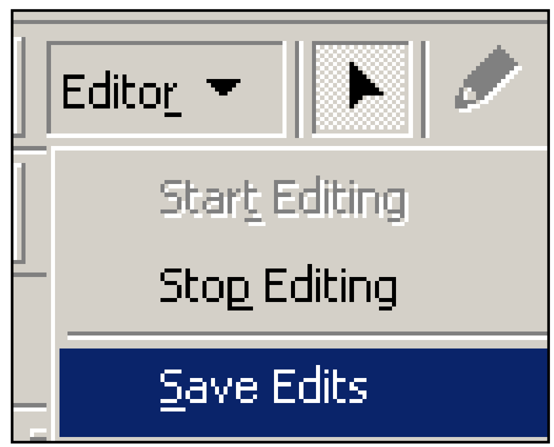
Start editing



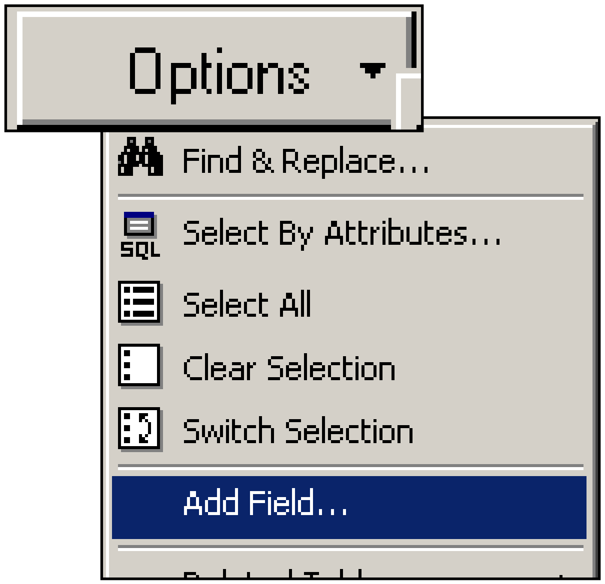
Type edits in fields

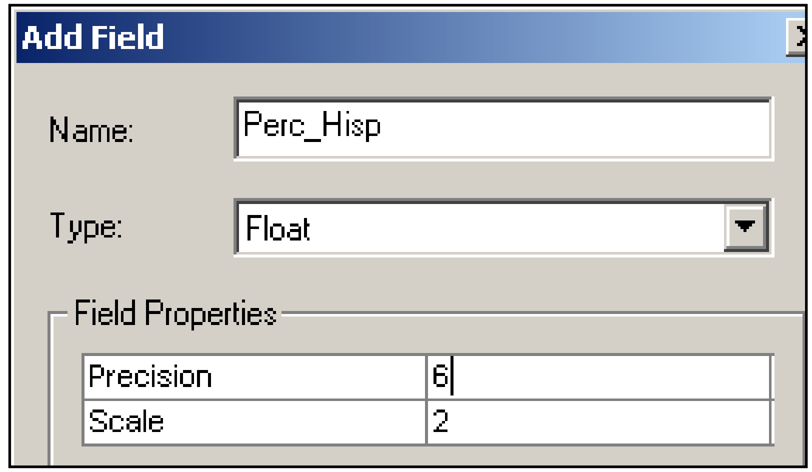


Save edits, stop editing



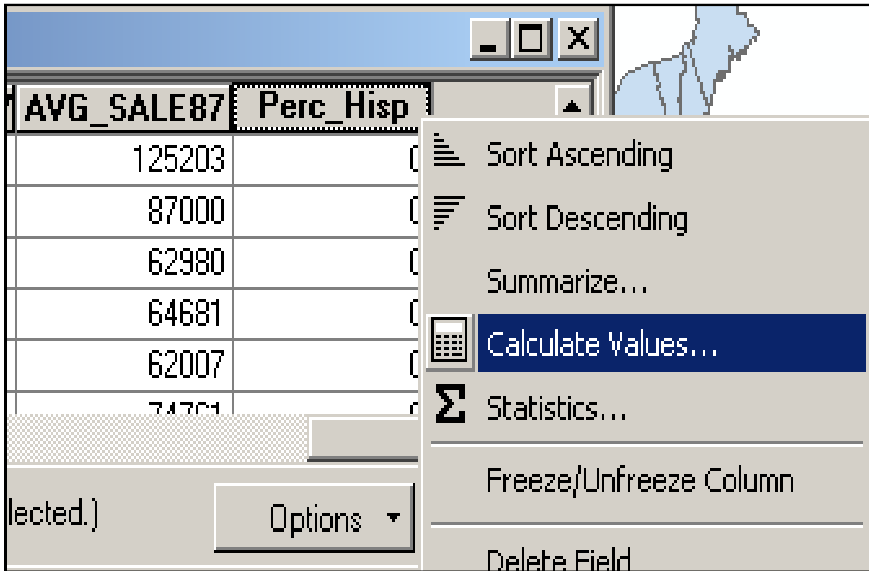
Calculating fields

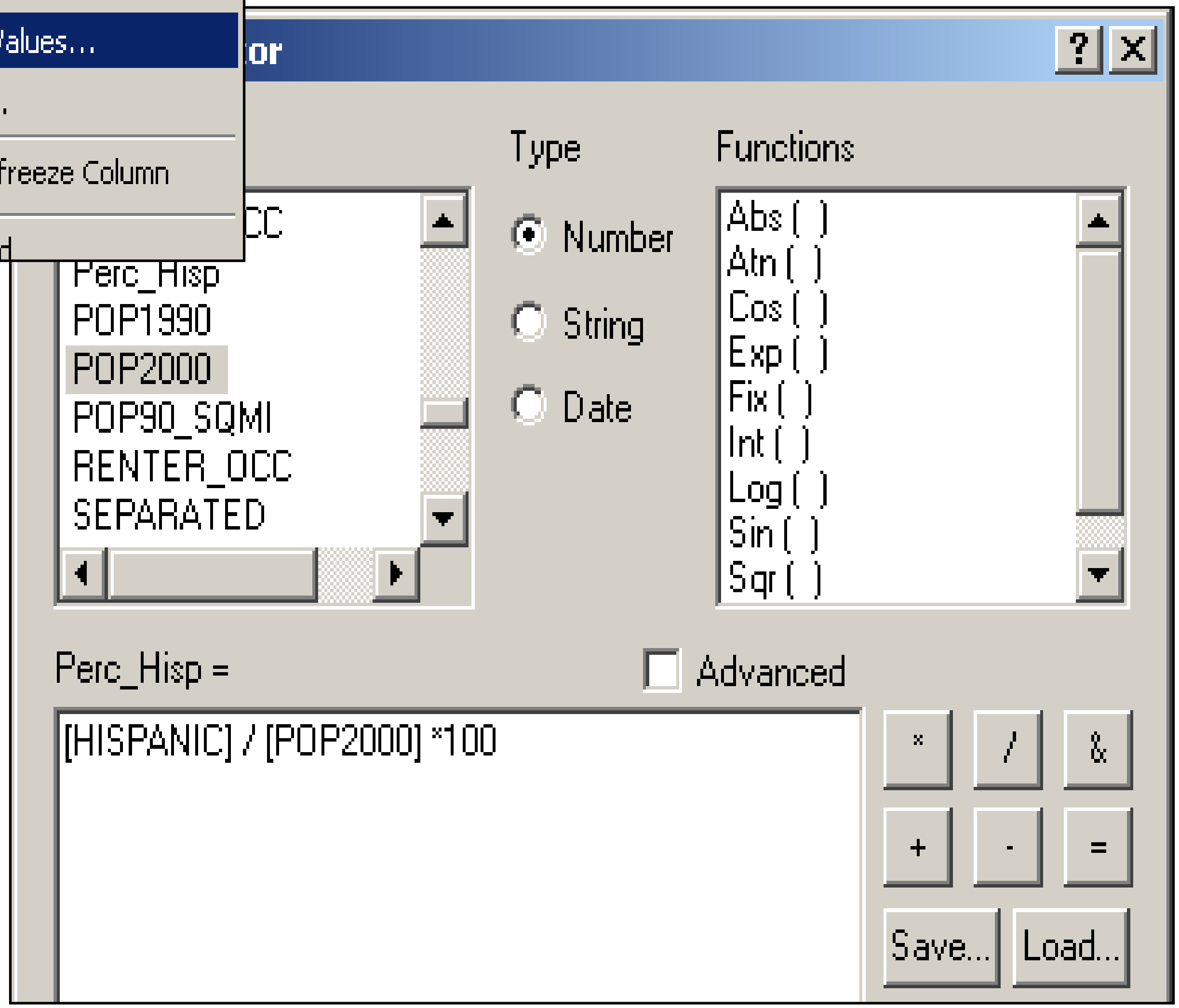




Add a new field if necessary
Consider whether you need decimal places!

Calculate





Right-click field to calculate

Enter expression

Chapter 5. Queries

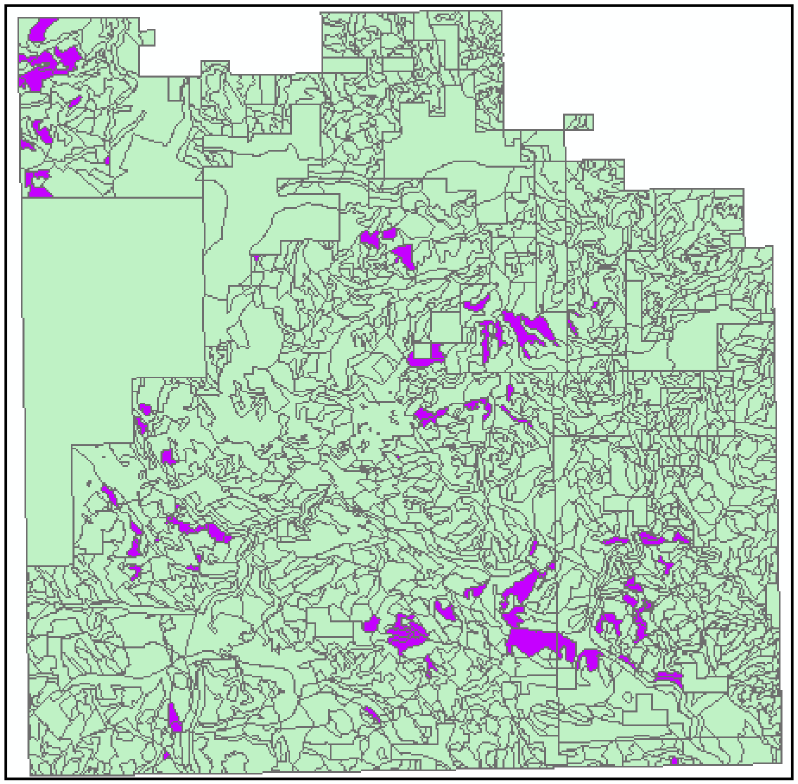
Objectives:

- Understanding **queries** and how they are used
- Selected features based on **attributes** using **SQL** and **Boolean** operators
- Selected features based on their **spatial location** with respect to other features
- Applying **selection options**, including the selectable layers and the selection method

What are queries?

- **Extract certain records** from a map or table
- Records **meet certain criteria**
  - **Aspatial queries**
    - All parcels with **value greater than $100,000**.
  - **Spatial queries**
    - All parcels that **lie completely within the flood plain**

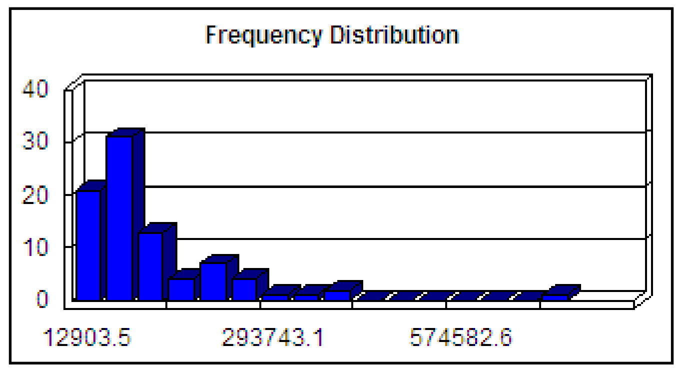
Selecting features of interest



Selecting aspen stands from a forest vegetation layer.

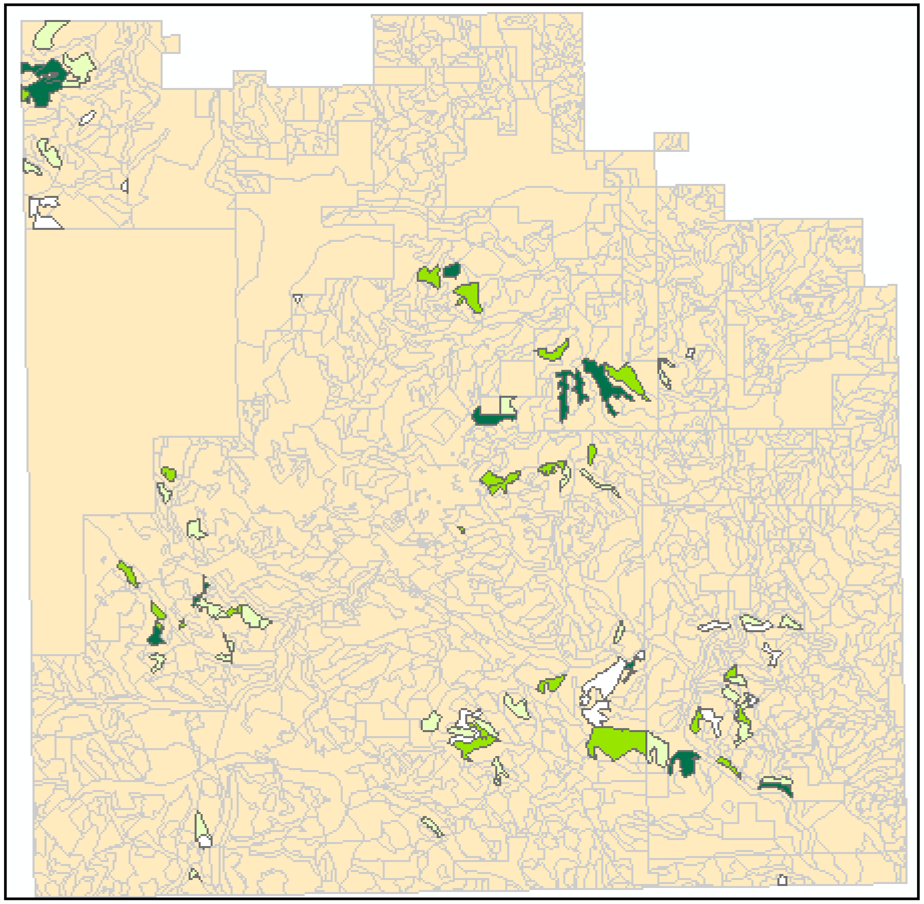
Using statistics on areas (m^2)

Minimum:	12,900
Maximum:	750,500
Sum:	10,529,000



[COV_TYPE] = “TAA”

Exploring patterns

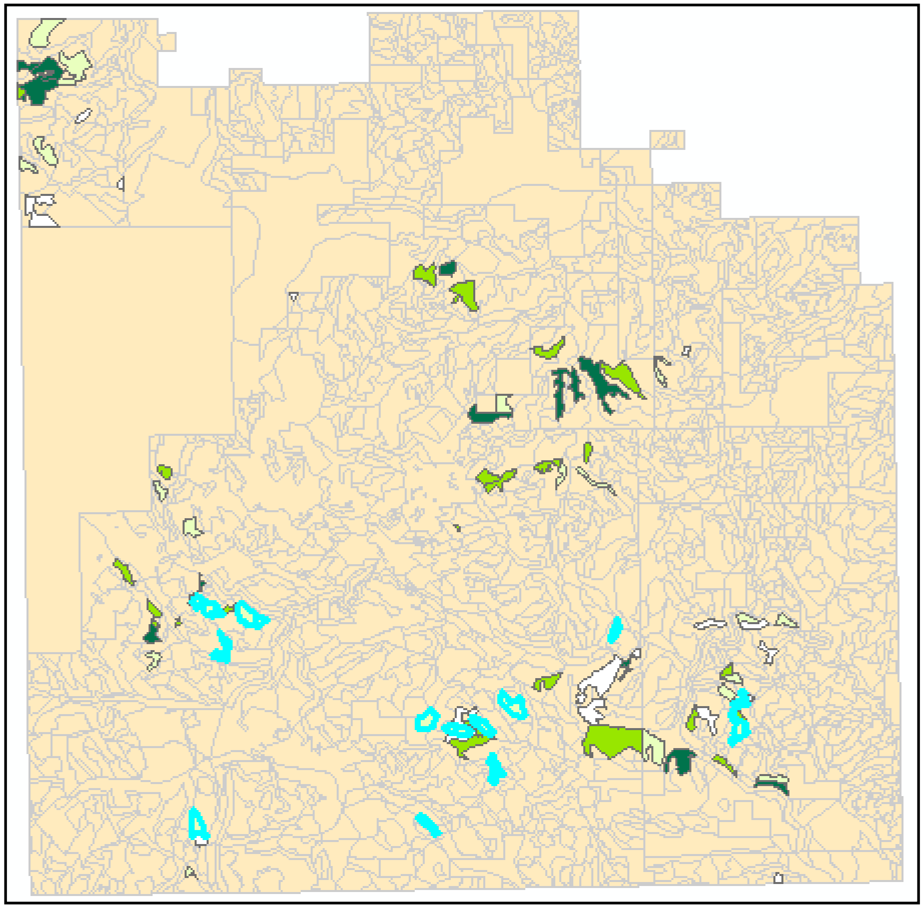


Are aspen stands **randomly scattered or clustered**?

Do they occur in **particular portions** of the forest?

What are the **distributions of stand densities**?

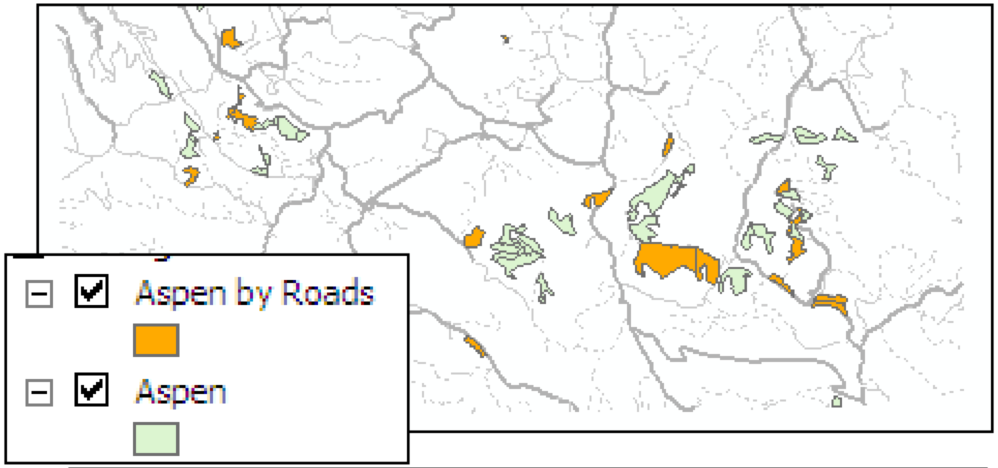
Isolating for more analysis



Are there any **mature stands with large trees and open crowns**? **Where** are they?

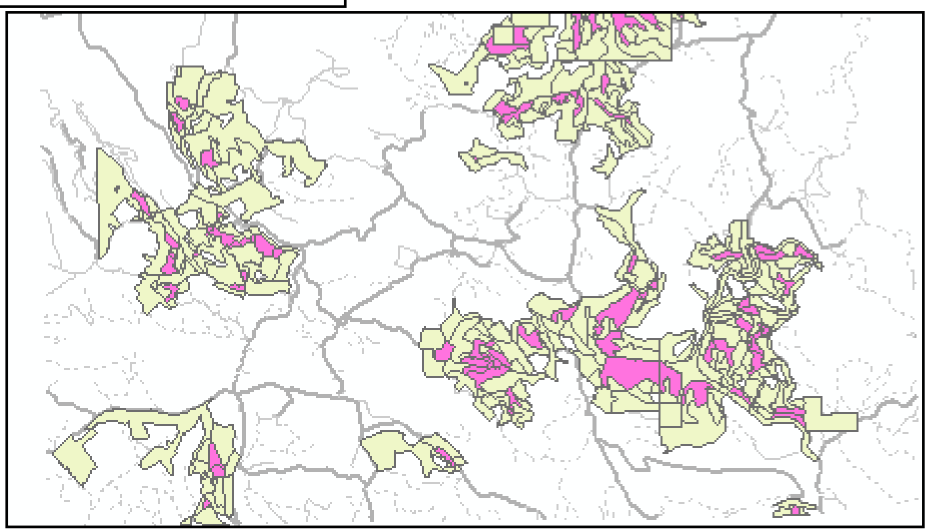
[TREE_SZ96] = 'L' AND [DENSITY96] = 'A'

Exploring spatial relationships



What fraction of stands are **intersected by roads**?

What **types of trees** are **adjacent** to aspen stands?



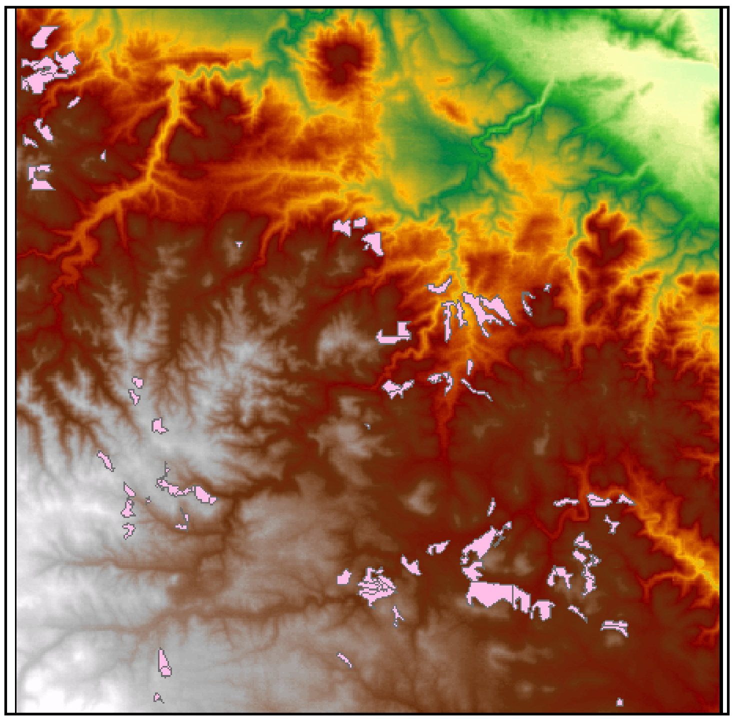
Attributes of nextaspenstats

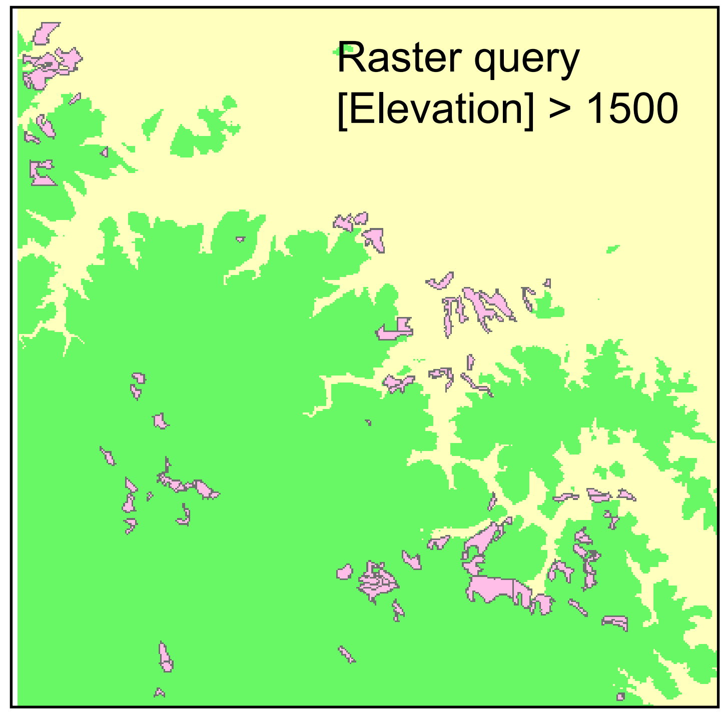
COV_TYPE	Count_	Sum_Shape_1_Area
TPP	236	41075786.751212
	33	18307652.679152
TAA	85	10529137.894304
TBO	5	1296700.771798
GRA	5	500328.341753
TWS	5	326514.674681
TLP	1	35111.344044
NFL	1	23186.765917

Queries involving surfaces

Over **what range of elevations** do aspen occur?

Do aspen occur **above 1500m** elevation?

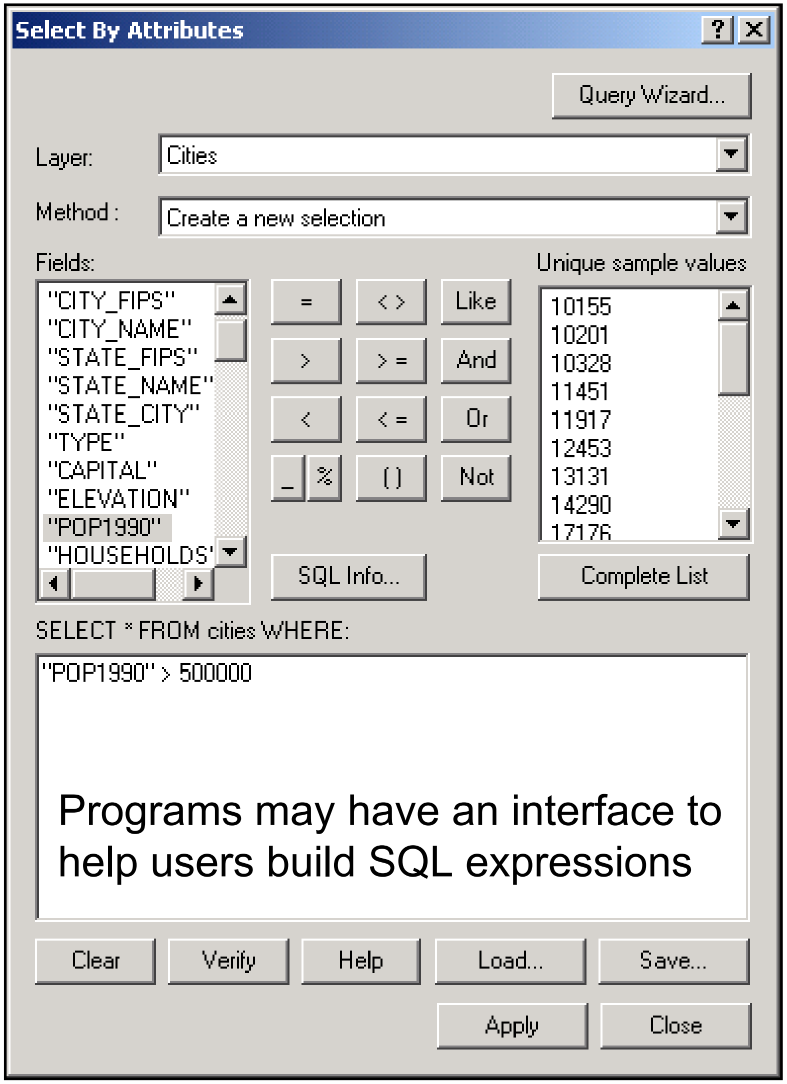




SQL

- Many databases use a **special query language** called **Structured Query Language**
- Can write queries that **work in multiple DBMS** environments
- Queries can be **saved and reused**
- Nearly always **case-sensitive**

SQL Query Examples



Some Valid Queries

SELECT *FROM cities WHERE "POP1990" >= 500000

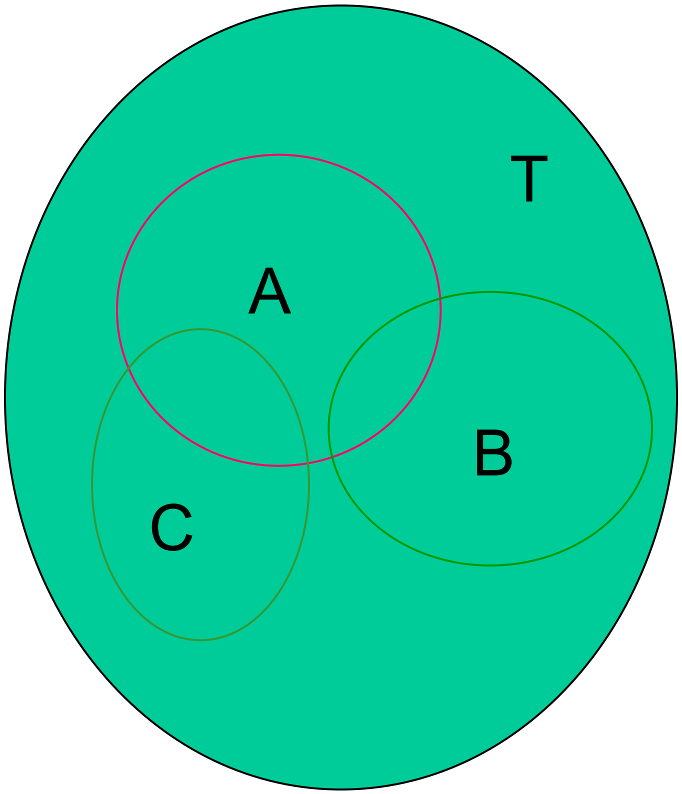
SELECT *FROM counties WHERE “BEEFCOW_92” < “BEEFCOW_87”

SELECT *FROM parcels WHERE “LU-CODE” = 42 AND “VALUE” > 50000

SELECT *FROM rentals WHERE “RENT” > 700 AND “RENT” < 1500

In most databases, SQL expressions are case-sensitive “Smith” ≠ “SMITH”

Queries as sets

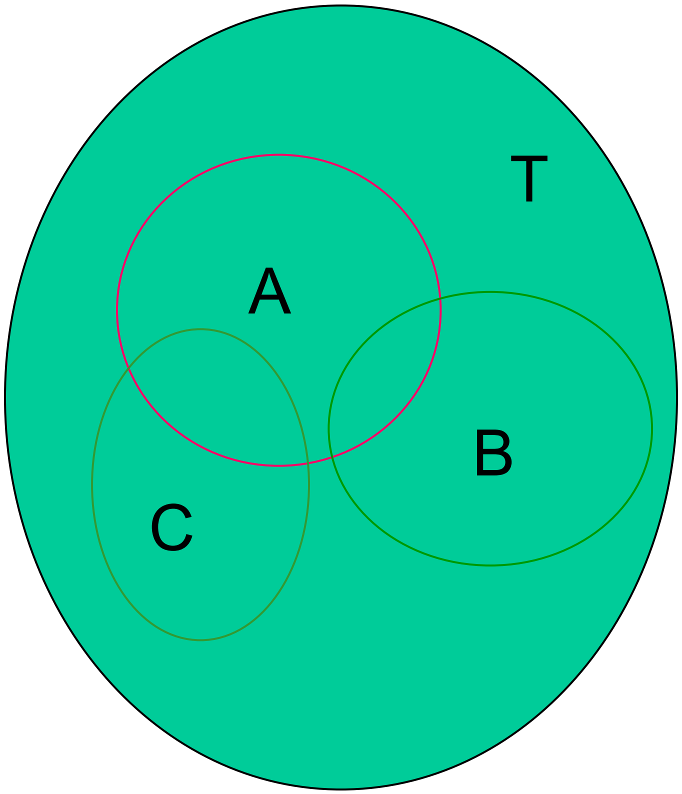


- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

Queries are used to **extract subsets** (records) of interest from a set (table).

Multiple criteria may be used (such as Geography majors from New York)

Single criteria

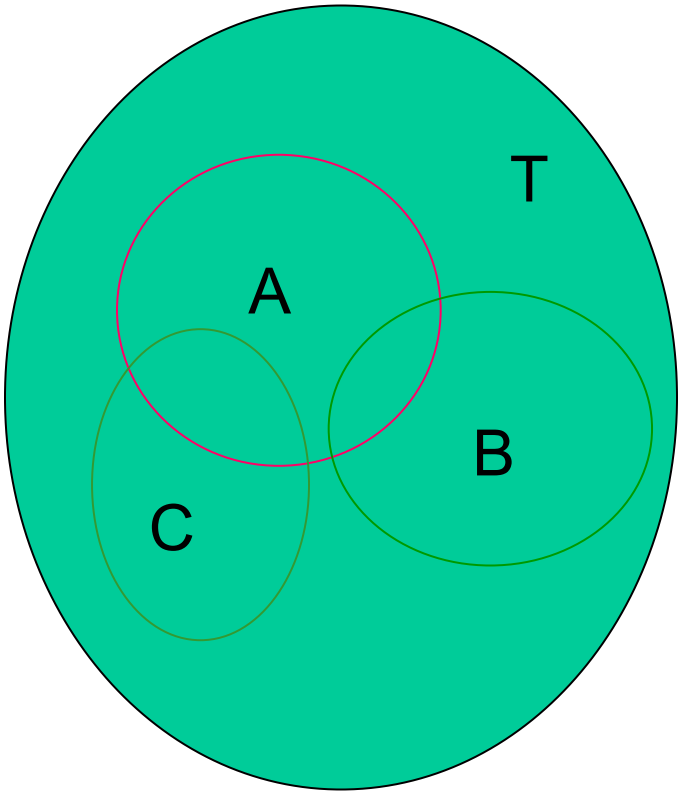


- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

Select students from T where [Home_State] = “NY”

Select students from T where [Major] = “Geography”

Double criteria

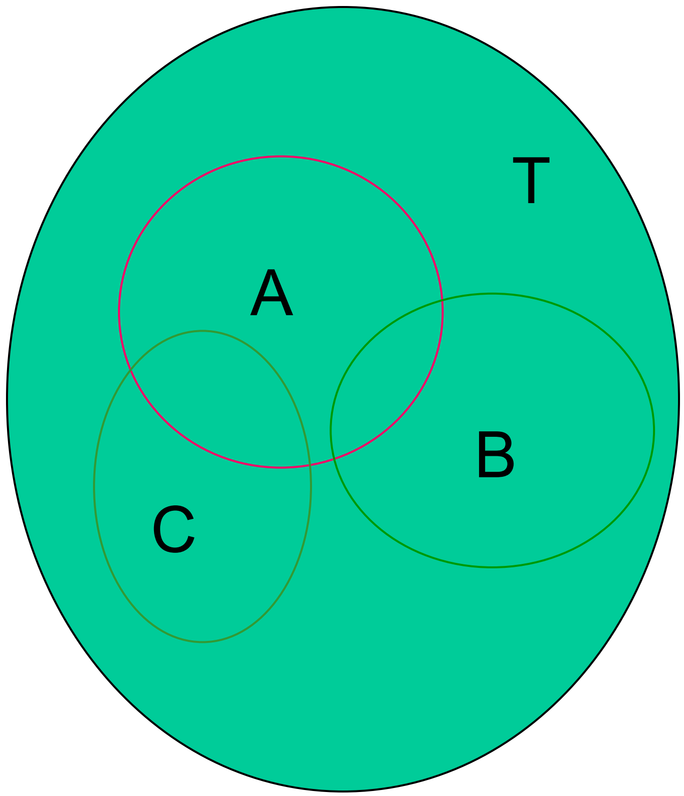


- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

Select students from T where
[Home_State] = “NY” OR [Home_State] = “NJ”

Select students from T where
[Home_State] = “NY” AND [Major] = “Geography”

AND vs OR?



- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

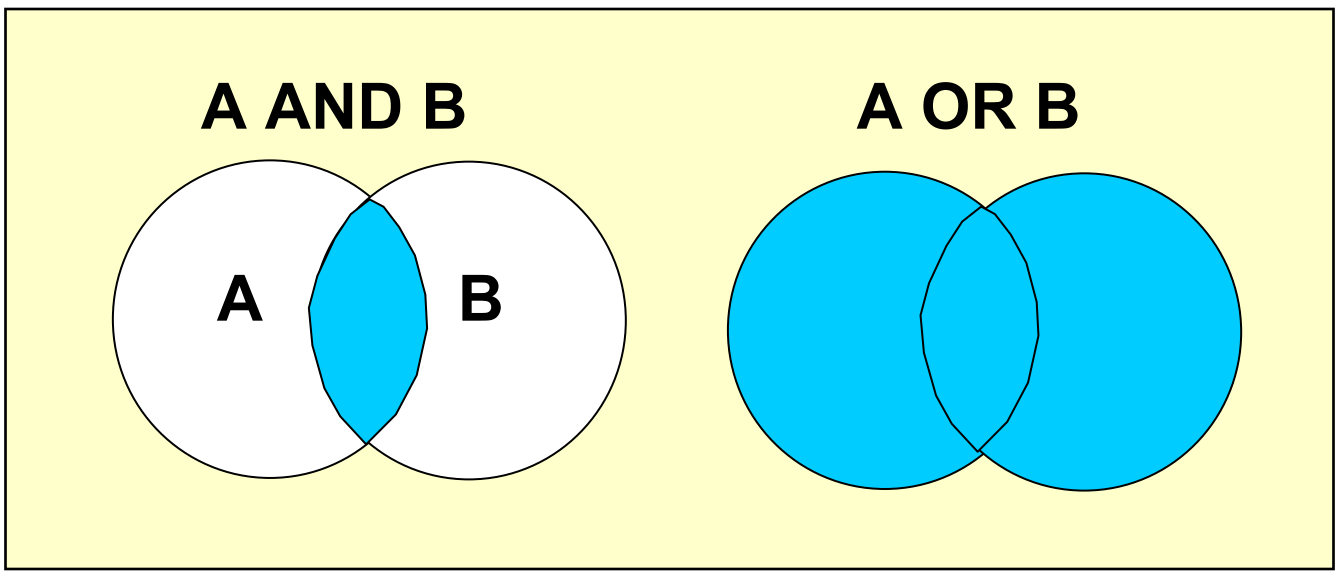
Select students from T where
[Home_State] = “NY” OR [Home_State] = “NJ”

Select students from T where
[Home_State] = “NY” AND [Major] = “Geography”

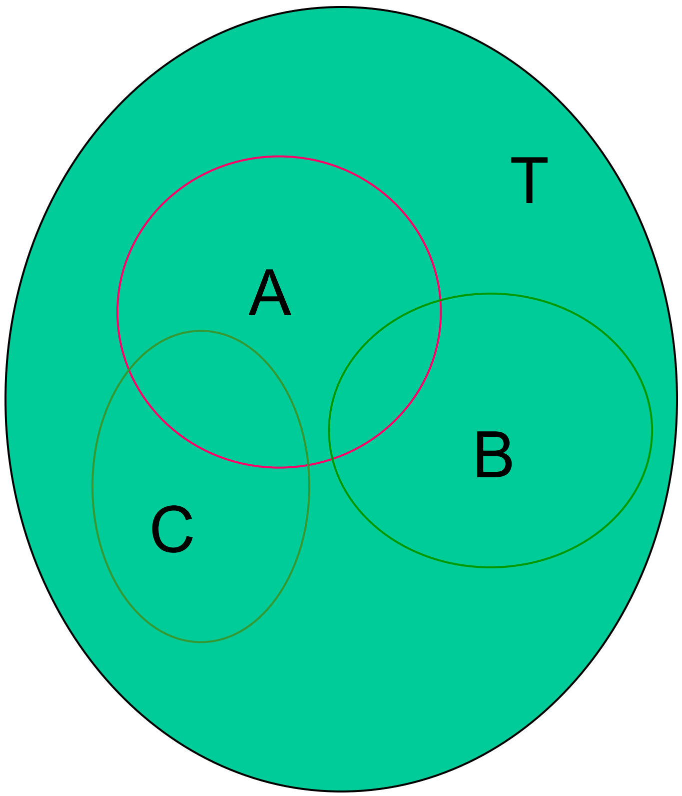
Each condition is **tested separately**. If **AND** is used, then **BOTH must be true**. If **OR** is used, then **either may be true**.

Boolean expressions

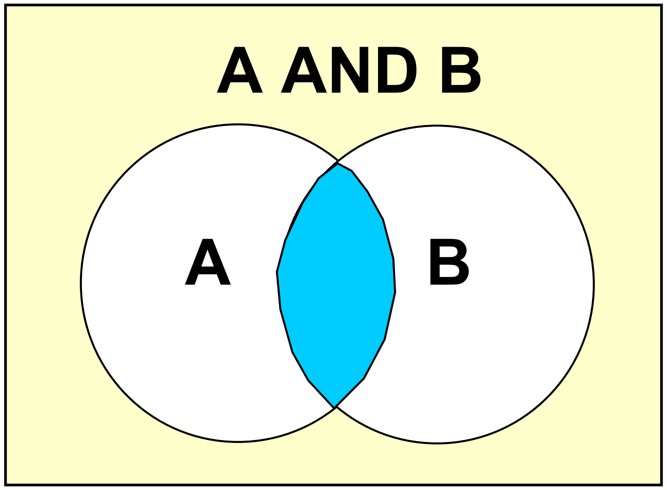
AND and OR are known as **Boolean operators**. Boolean operators are used to **evaluate pairs of conditions**.



AND vs OR?

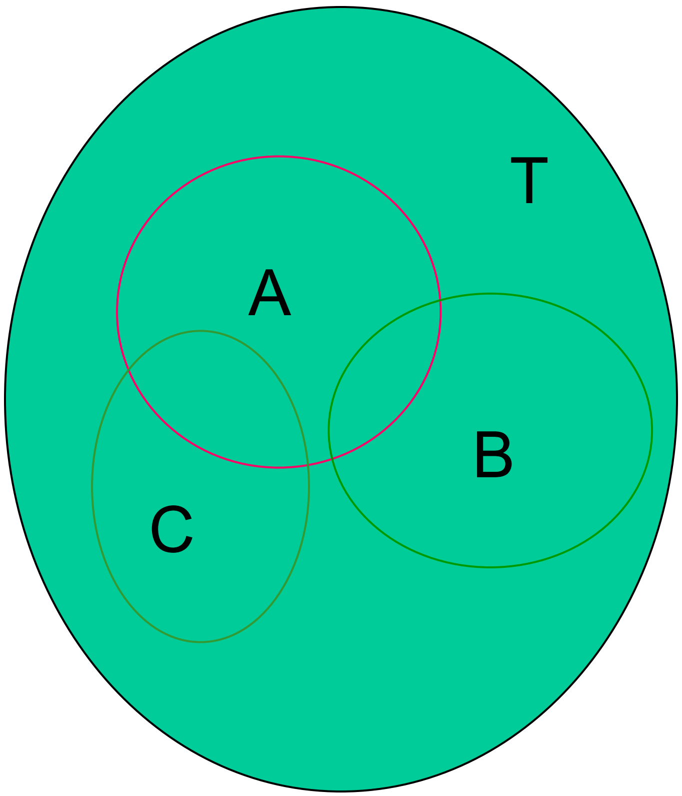


- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

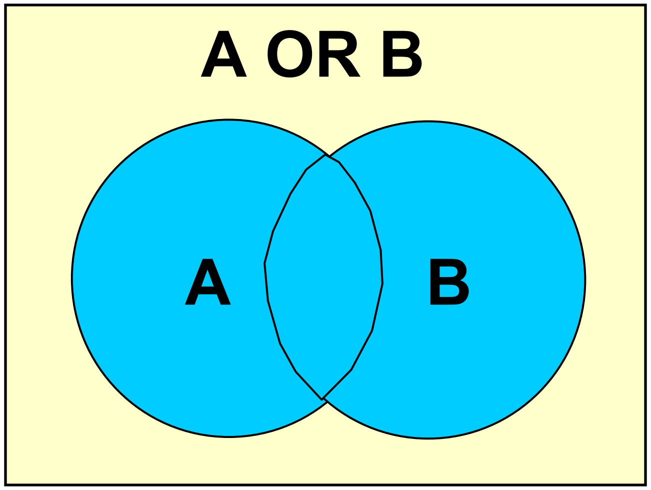


Select students from T where
[Home_State] = “NY” AND [Major] = “Geography”

AND vs OR?

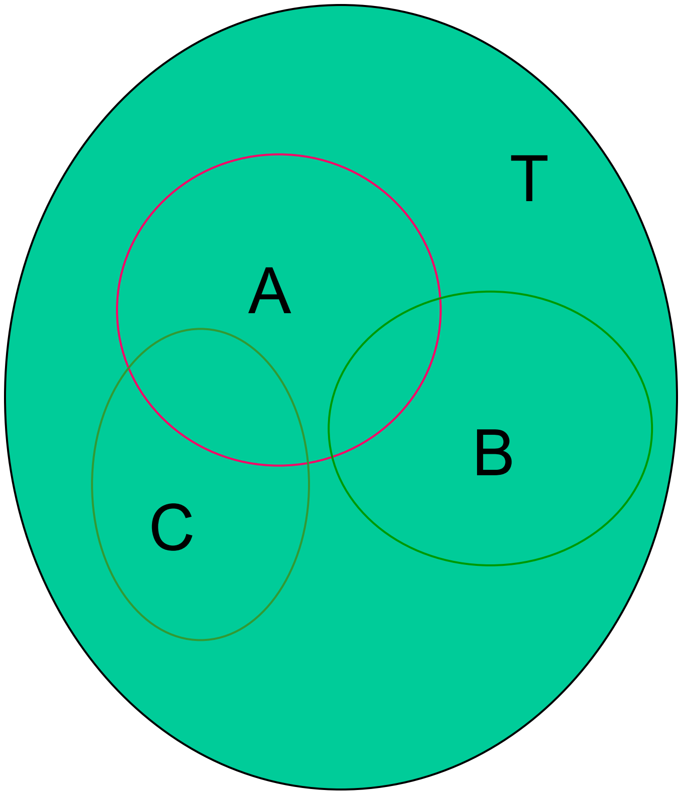


- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]



Select students from T where
[Home_State] = “NY” OR [Major] = “Geography”

What do you get?



- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

•Select students from T where
[Major] = “Geography” AND [Major] = English”

- B AND C

•Select students from T where
[Major] = “Geography” OR [Major] = “English”

- B OR C

•Select students from T where
[State] = “NY” AND [Major] = “English”

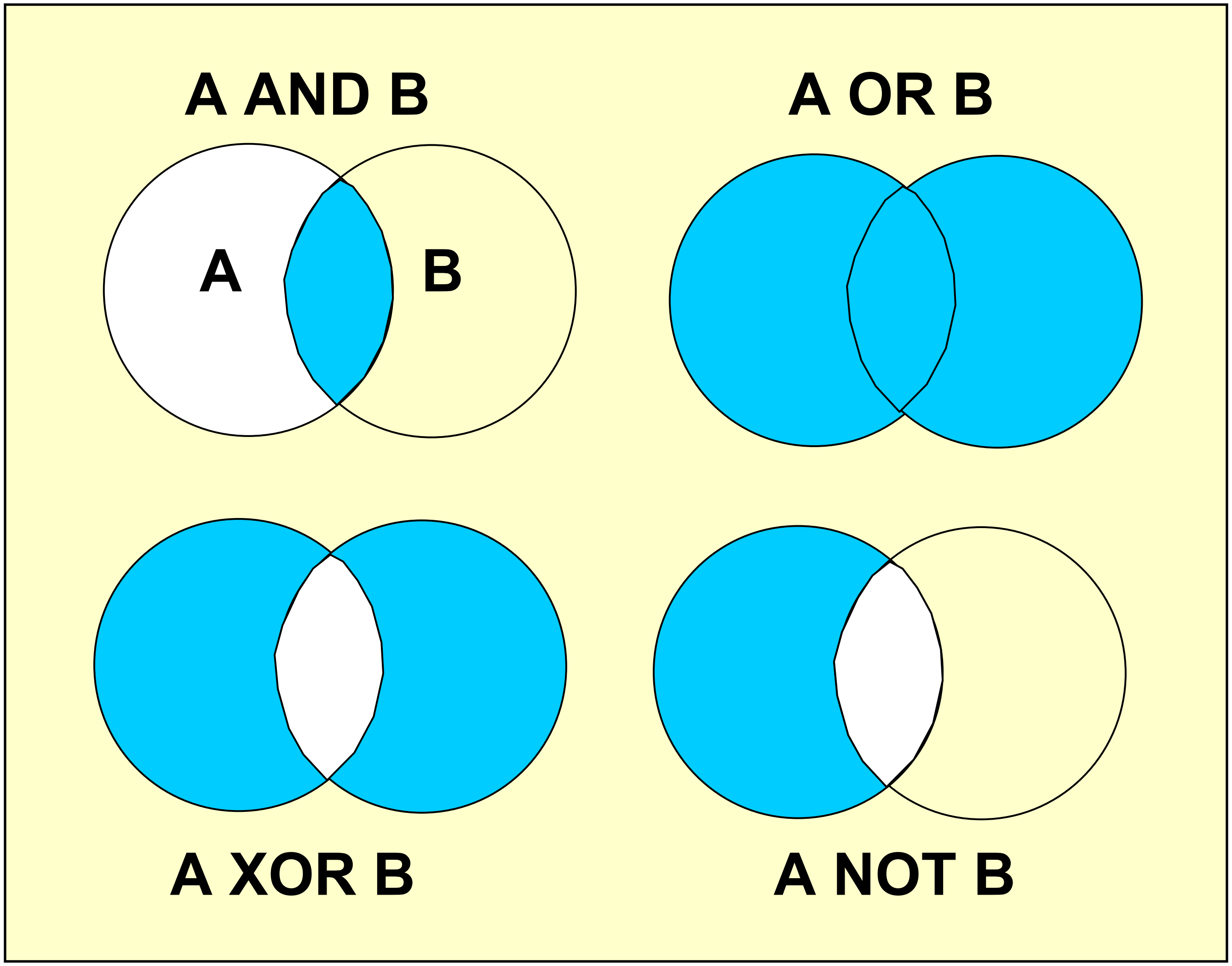
- A AND C

•Select students from T where
[State = “NY” OR [Major] = “English”

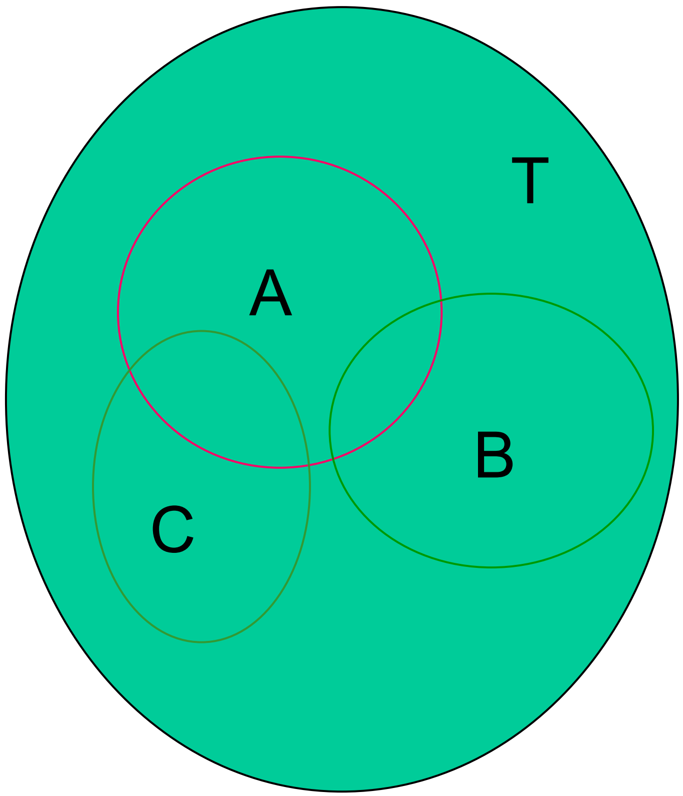
- A OR C

Other Boolean operators

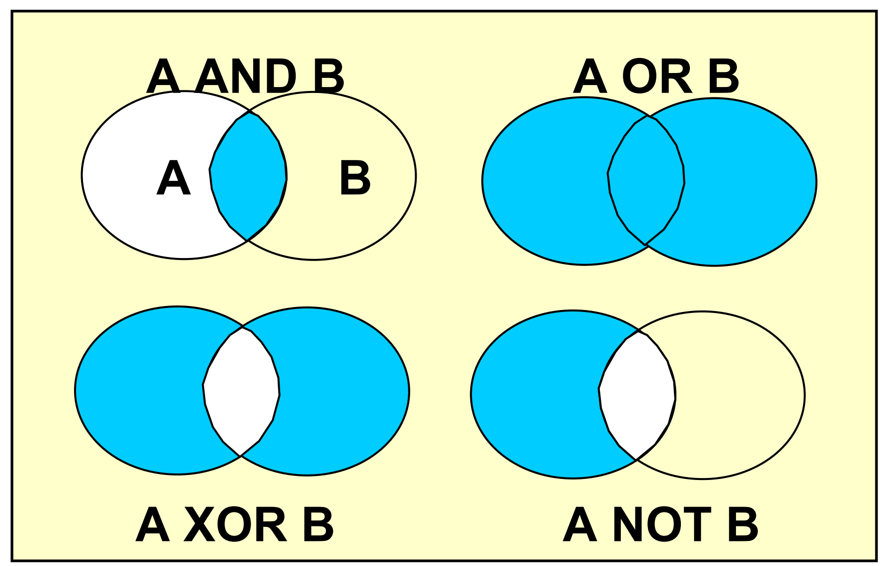
Some databases use additional operators besides AND and OR.



What do you get?



A AND B	B AND A
A OR B	B OR A
A XOR B	B XOR A
B XOR C	B XOR C
A NOT B	B NOT A
B NOT C	C NOT B



- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

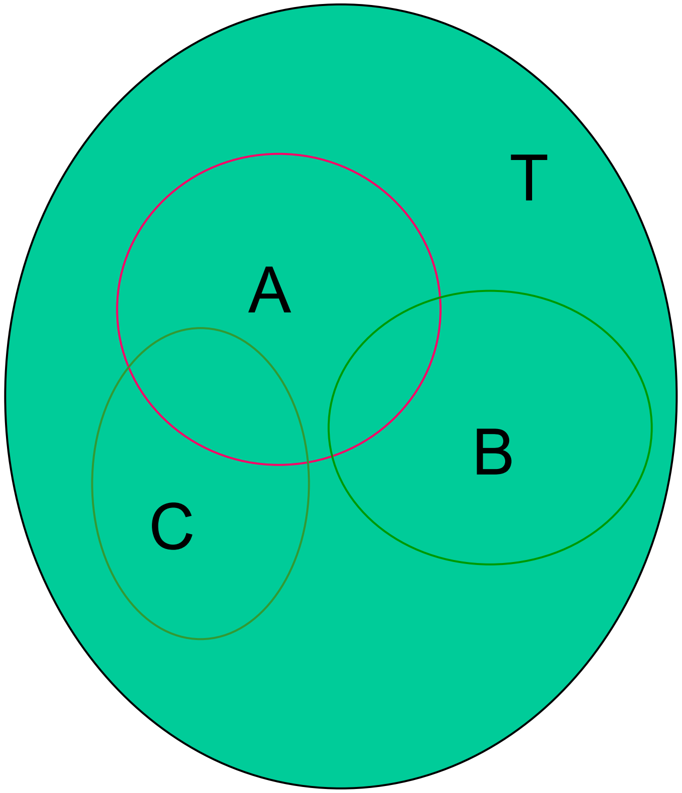
Commutation of operators

- AND, OR and XOR are **commutative**
  - A AND B == B AND A
  - A OR B == B OR A
  - A XOR B == B XOR A
- NOT is **not commutative**
  - A NOT B ≠≠ B NOT A

Order of operations

- Boolean operators have **equal order or precedence**
- Evaluation occurs from **left to right**
- **Parentheses** must be used to **change order**

What do you get?



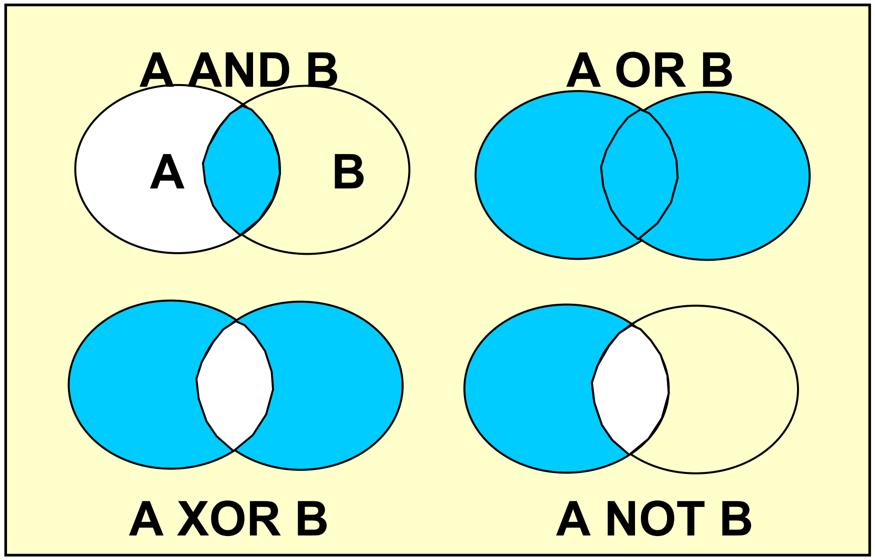
A AND B OR C

(A AND B) OR C

A AND (B OR C)

(A OR B) AND C

A OR (B AND C)



- Let T = [all students in University]
- Let A = [students from New York]
- Let B = [Geography majors]
- Let C = [English majors]

Searching for partial matches

- Sometimes you need to find **one string within another** rather than an exact match
  - Find all customer names **beginning with "Mac" or "Mc"**
  - Find all zip codes **beginning with 0**
- Typically uses a **"wildcard"** character
  - *Mac* or *Mc*
  - 0*

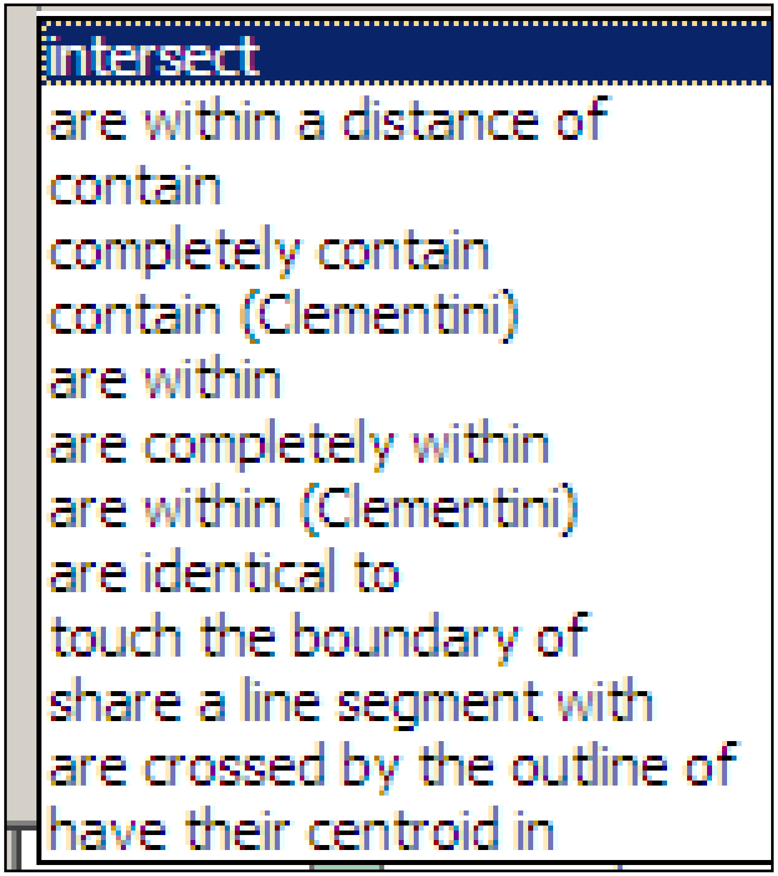
The Like Operator

- “NAME” **LIKE** ‘%(D)%’
  - Finds all of the (D) Democrats
- **%** is (single character) **wildcard**
- **Ignores** Don or Danforth

- “NAME” **LIKE** ‘%New %’
  - Would find New Hampshire and New York, but not Newcastle or Kennewick

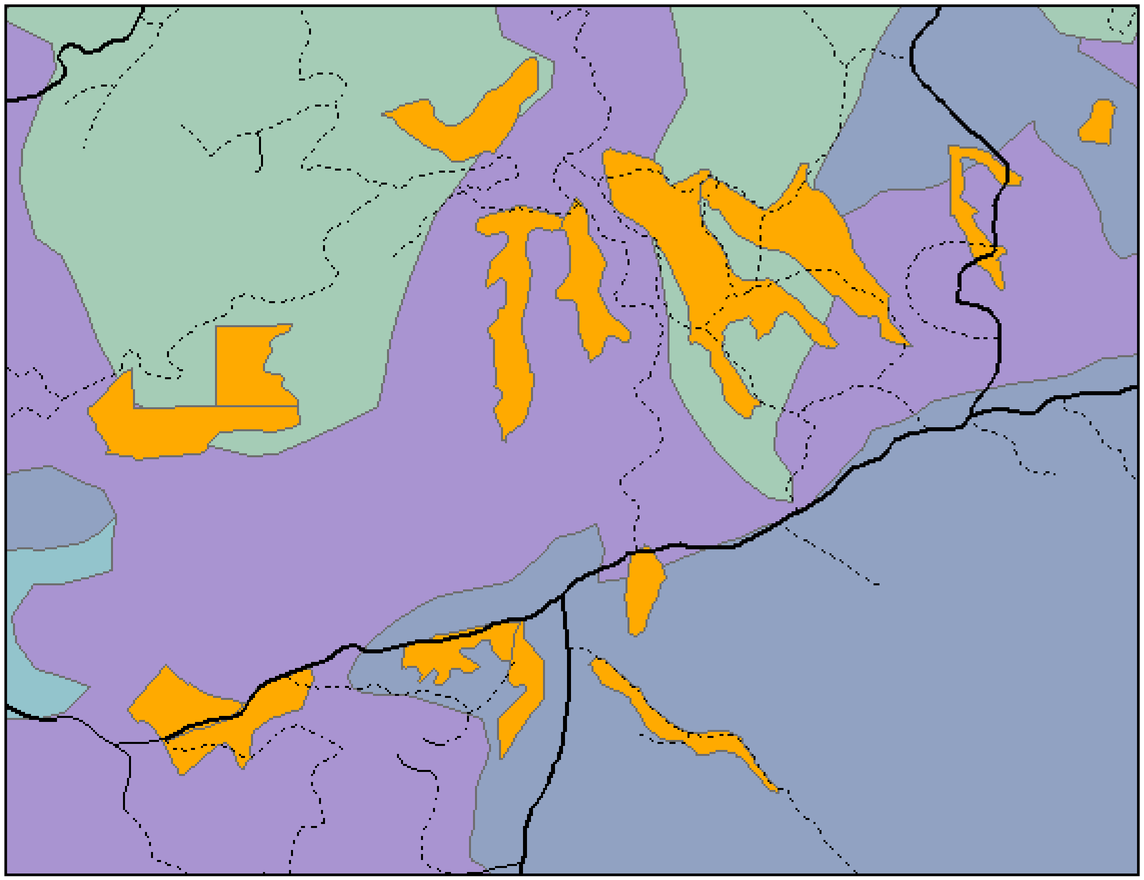
NAME
Sonny Callahan (R)
Terry Everett (R)
Bob Riley (R)
Robert B. Aderholt (R)
Robert E. "Bud" Cramer, Jr. (D)
Spencer Bachus (R)
Earl F. Hilliard (D)
Don Young (R)
Matt Salmon (R)
Ed Pastor (D)
Bob Stump (R)
John Shadegg (R)
Jim Kolbe (R)

Spatial operators



- **Spatial queries** can employ a number of **operators** to test the basic conditions of **intersection, containment,** and **proximity**.

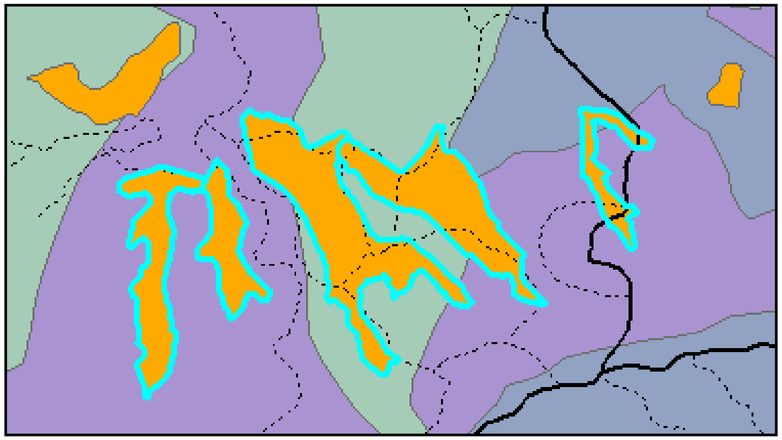
Basic spatial relationships



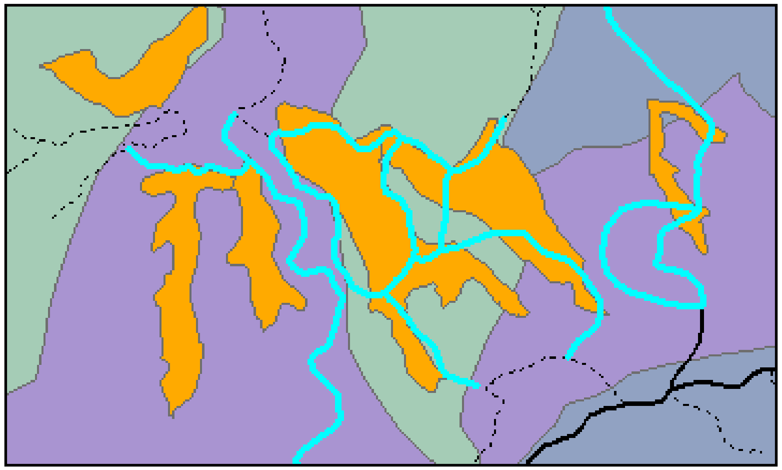
- **Intersection**
  - Does the road **cross** the aspen?
  - Do two polygons **share areas or boundaries**?
- **Containment**
  - Is the aspen **inside** a geology unit?
  - Is a road **inside** a geology unit?
- **Proximity**
  - How many aspen stands **within 200 meters** of a road?

- The operators **test relationships** between **two layers at a time**.
  - The **target** layer is the one containing the features to be selected
  - The **source** layer is the one containing the features being compared to.

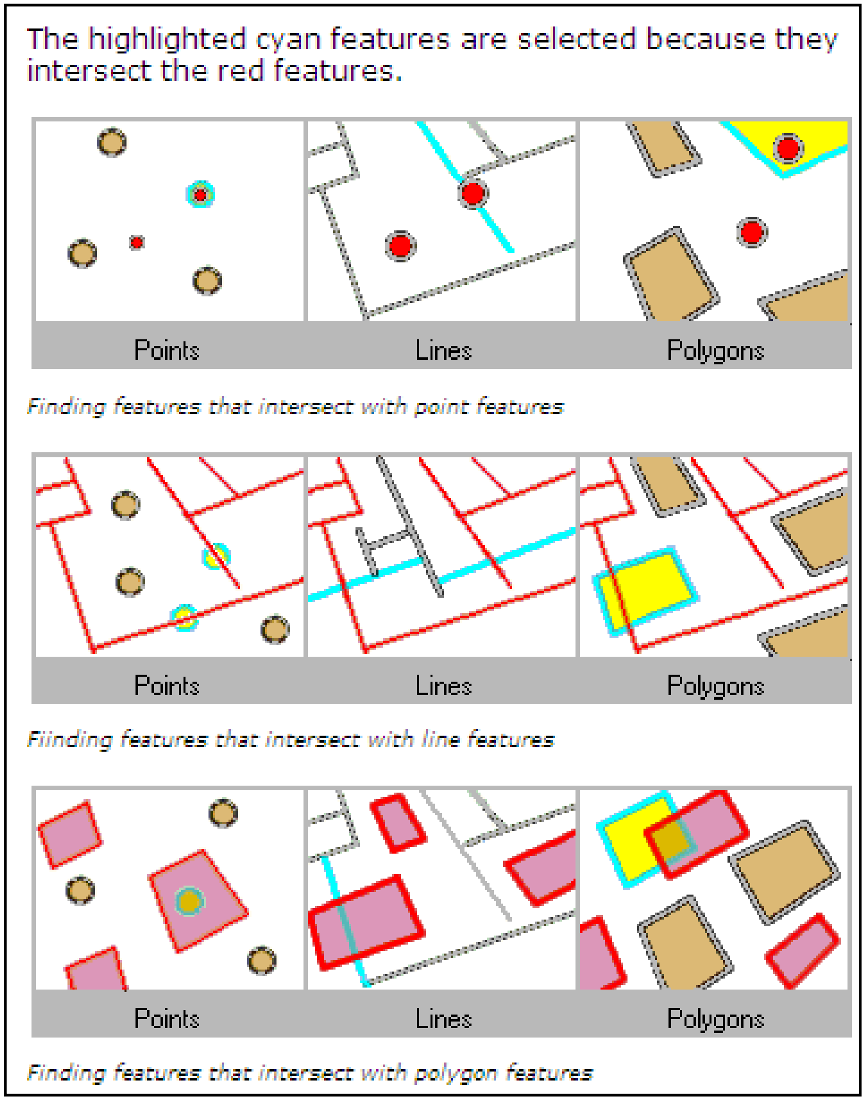
Select the **aspen** stands that are intersected by **roads**.



Select the **roads** that are intersected by **aspen** stands.

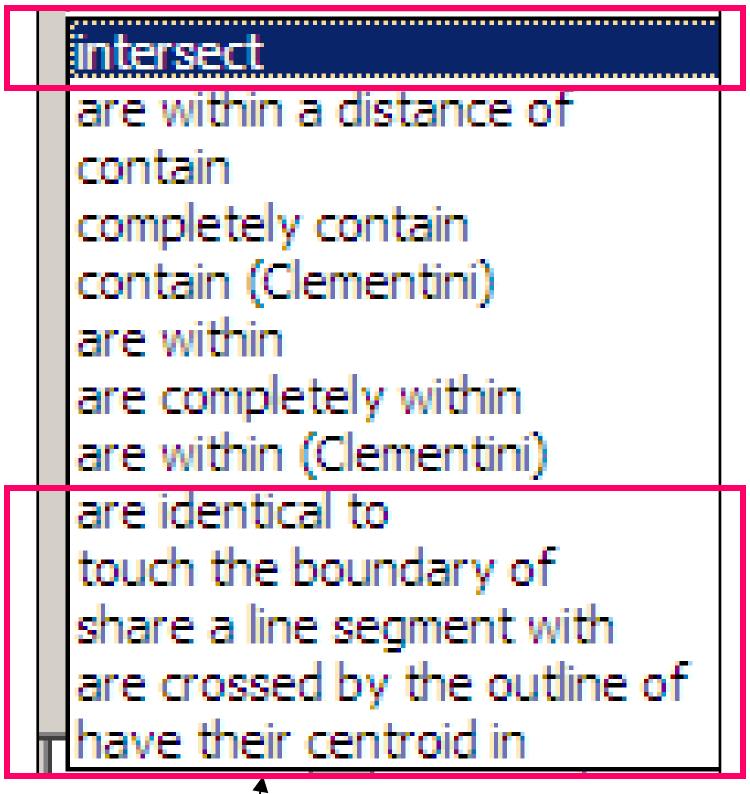


Spatial operators

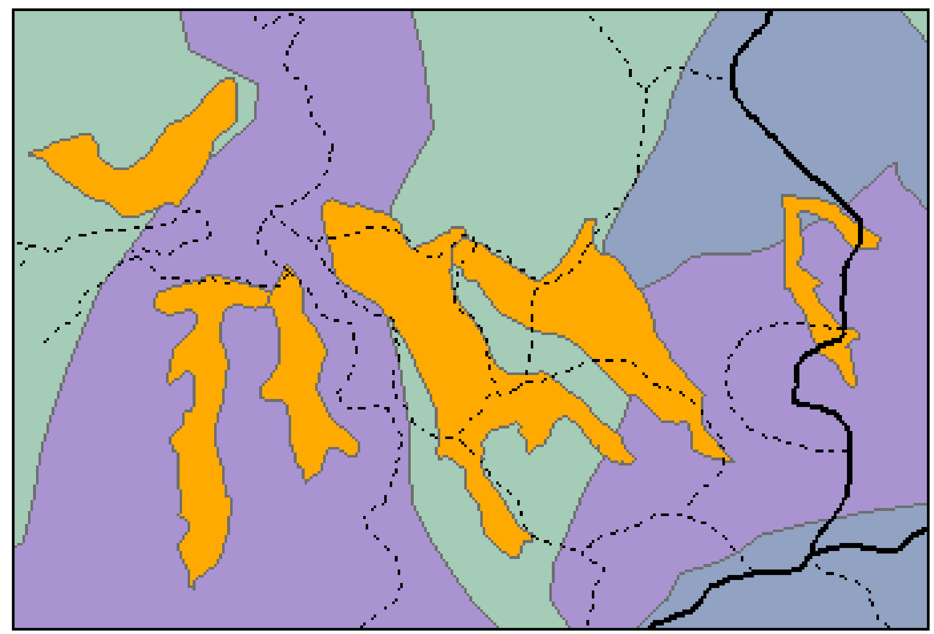


- The use and action of the operators **depends on the feature geometry** (points, lines polygons)
- Some operators can only be used with **certain geometry types**
  - Polygons can contain points, but not vice versa

Intersection operators

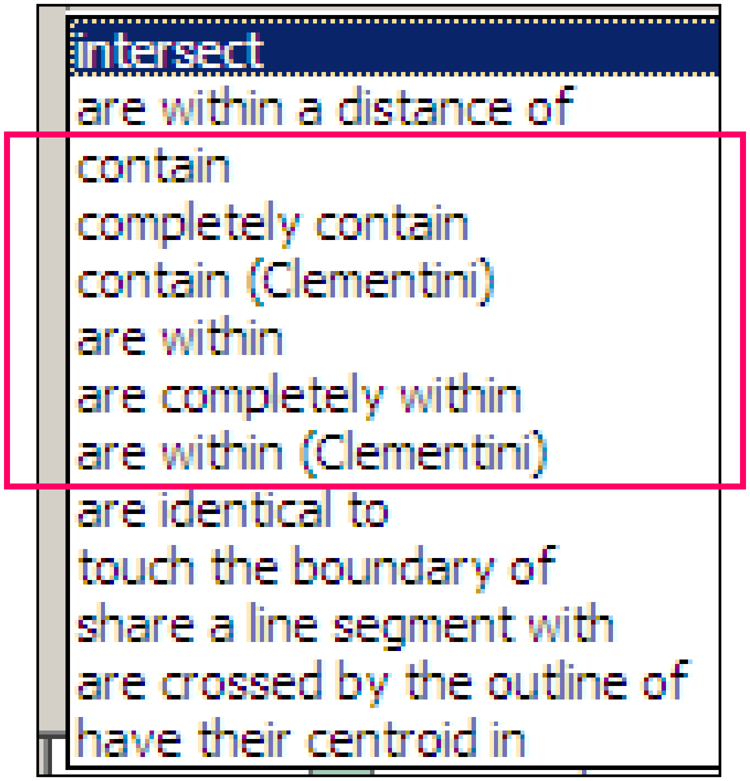


- Features **intersect** when any part of one feature touches, crosses, or overlaps another feature.

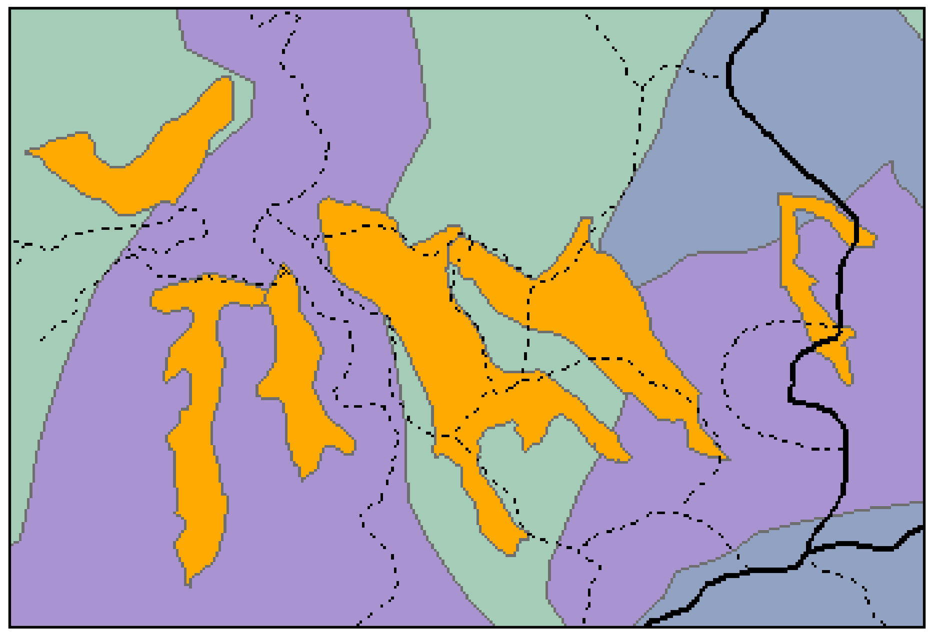


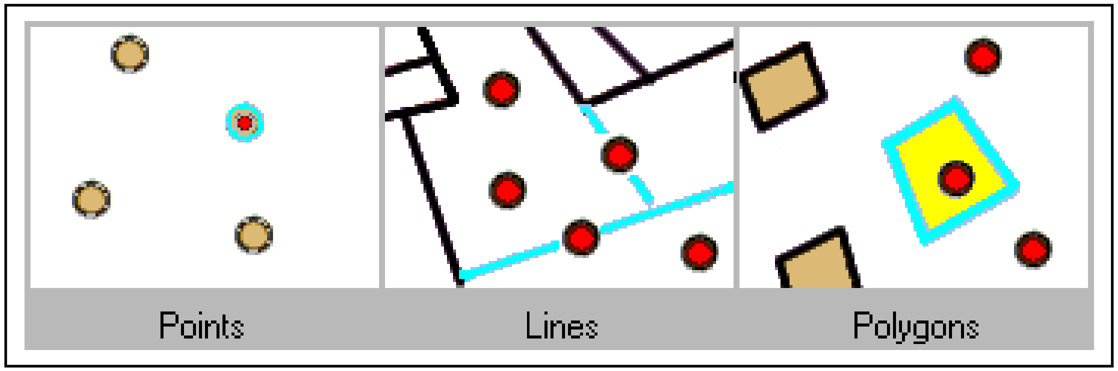
The lower set includes "special cases" of intersecting features.

Containment operators

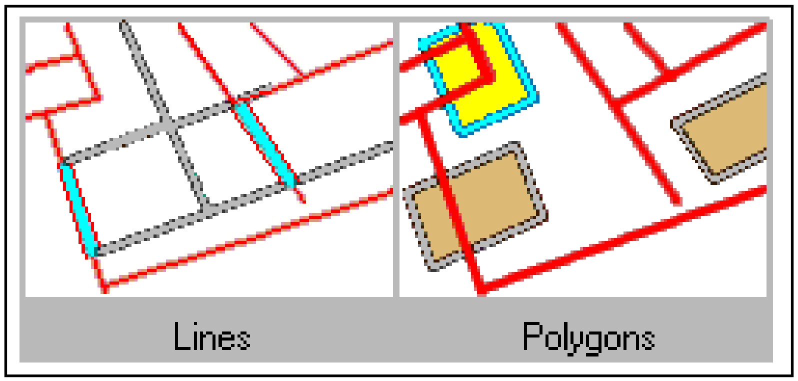


- Features that **enclose** all of another feature **contain** it.
- **Within** is the **inverse** of **contain**

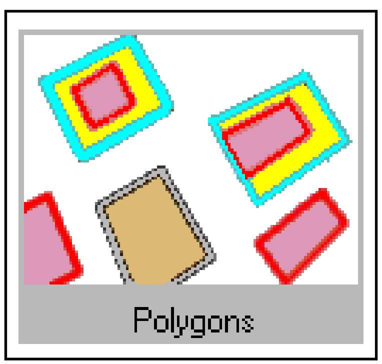




Selecting features that contain points



Selecting features that contain lines



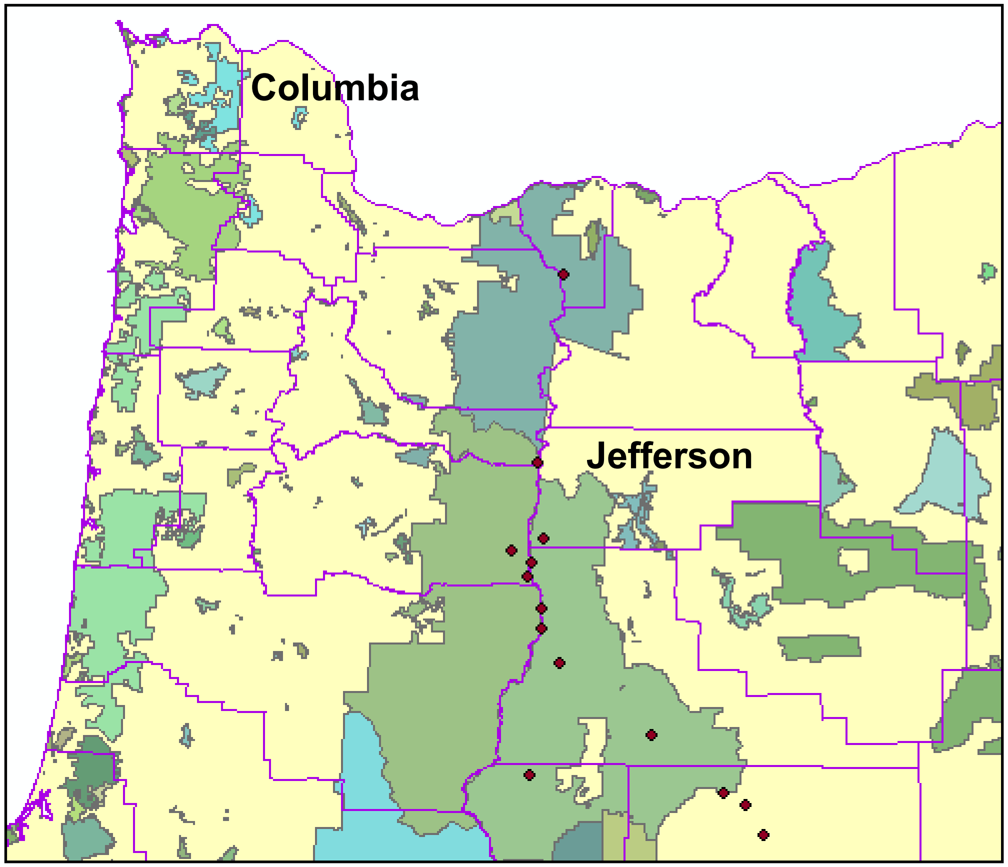
Selecting features that contain polygons

Containment operators are **affected by geometry type**.

Notice that **only polygons** can **contain other polygons**.

However, points can contain other points, lines can contain points or lines, and polygons can contain anything.

Types of containment



- **Contains**
  - One feature lies inside another and may share a boundary
  - Oregon contains Columbia county
- **Completely contains**
  - One feature lies inside another without touching the boundary
  - Oregon does not completely contain Columbia county, but does completely contain Jefferson county

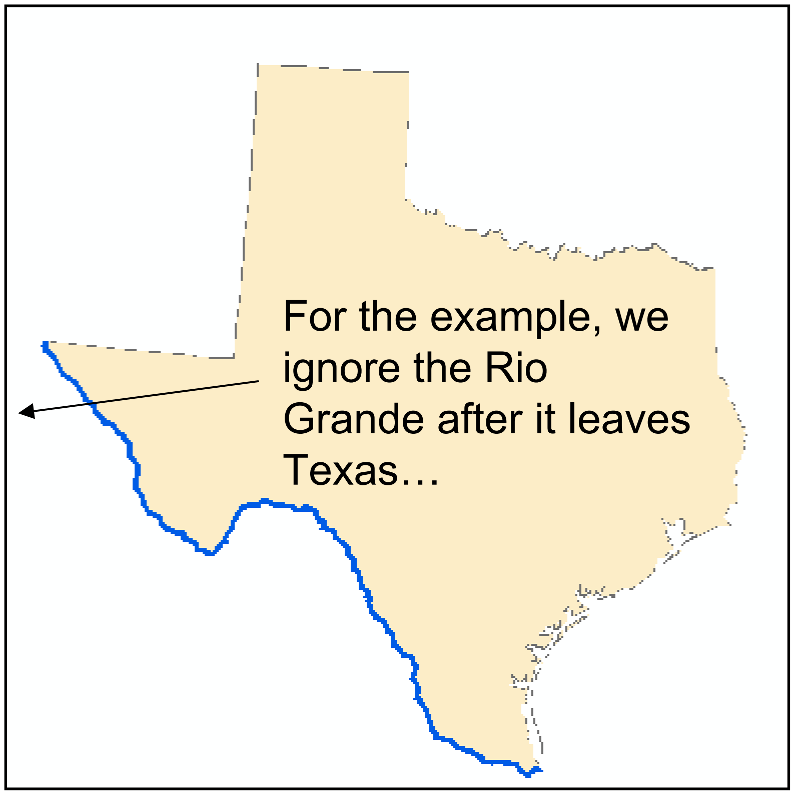
Within is the inverse. Columbia county is within Oregon. Jefferson county is completely within Oregon.

Clementini operators

- **Eliseo Clementini** and his coauthors defined a **special set of topological relationships** concerning containment*.
- Clementini **considers the boundary** of a polygon to be **separate from its inside or outside**.
- The Clementini operator is **equivalent to the standard operator except** when the **source feature lies only** on the **boundary of the target feature.**

*Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom, A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Proceedings of the Third International Symposium on Advances in Spatial Databases, pp. 277-295, June 23-25, 1993.

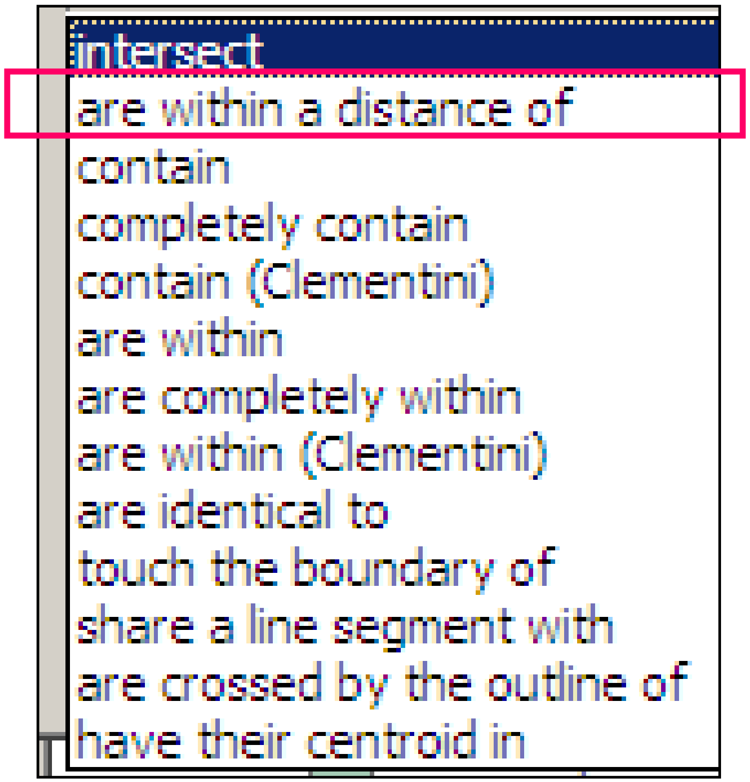
Clementini example



- The Rio Grande River lies **on the border of Texas**
  - The **Contains** operator **would select** the Rio Grande
  - The **Clementini Contains** operator **would NOT select** the Rio Grande because the state boundary is not considered part of Texas

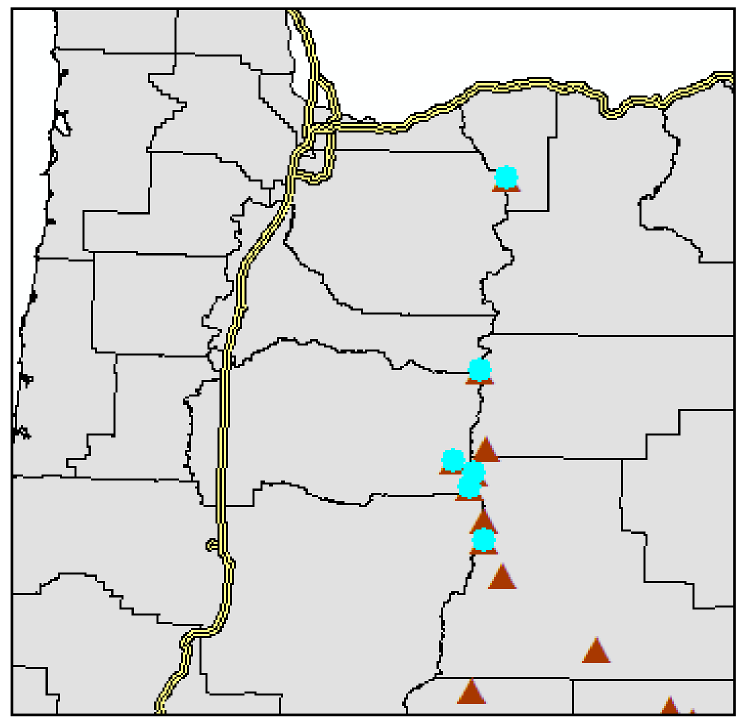
Conversely, the Rio Grande is within Texas using the standard operator, but is not within Texas using the Clementini operator.

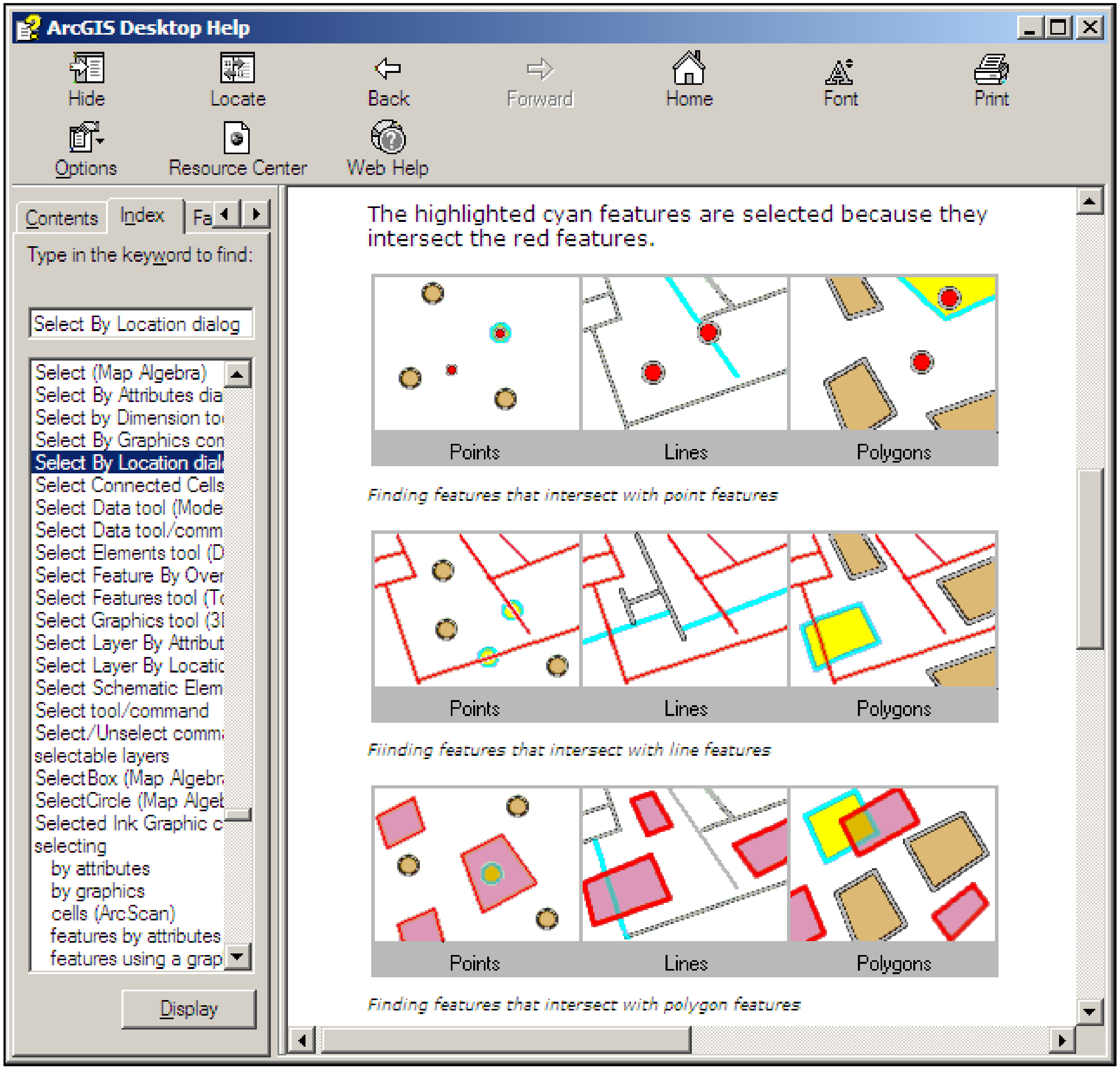
Proximity operators



- This operator tests whether the target features are **within a specified distance** of the source features.

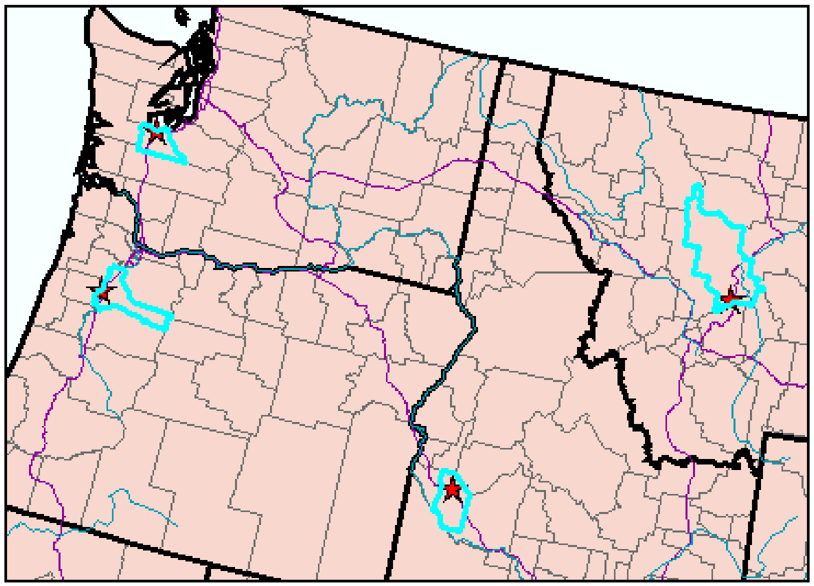
Volcanoes within 100 km of an interstate



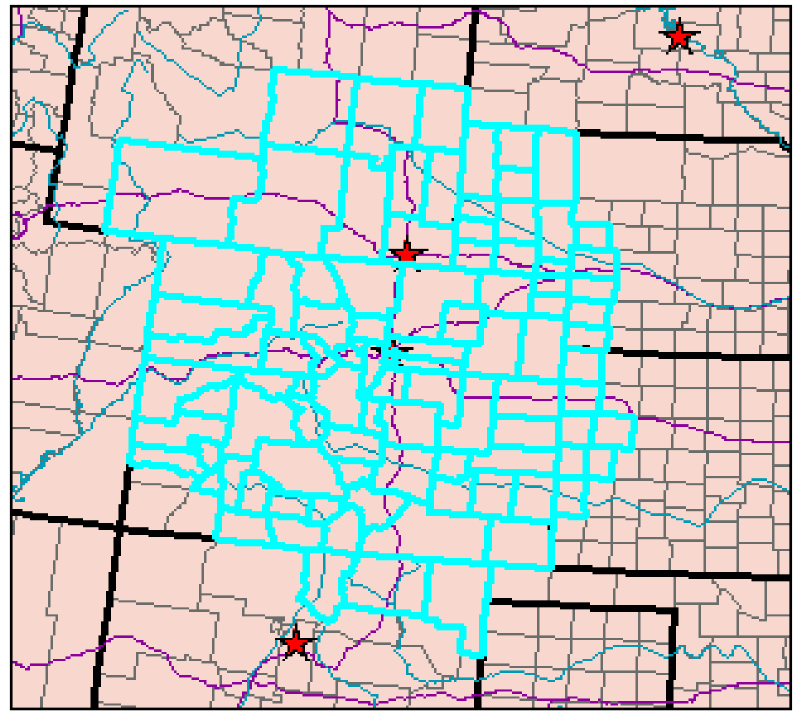


For more information on each operator and how they apply in the case of points, lines, and polygons, consult the Help information for the Select By Location command in ArcGIS.

More examples

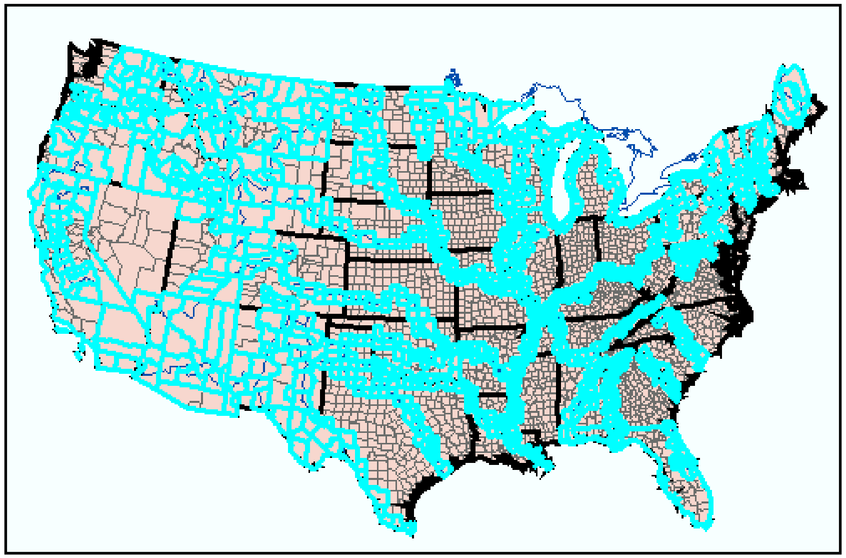


Select counties that contain state capitals

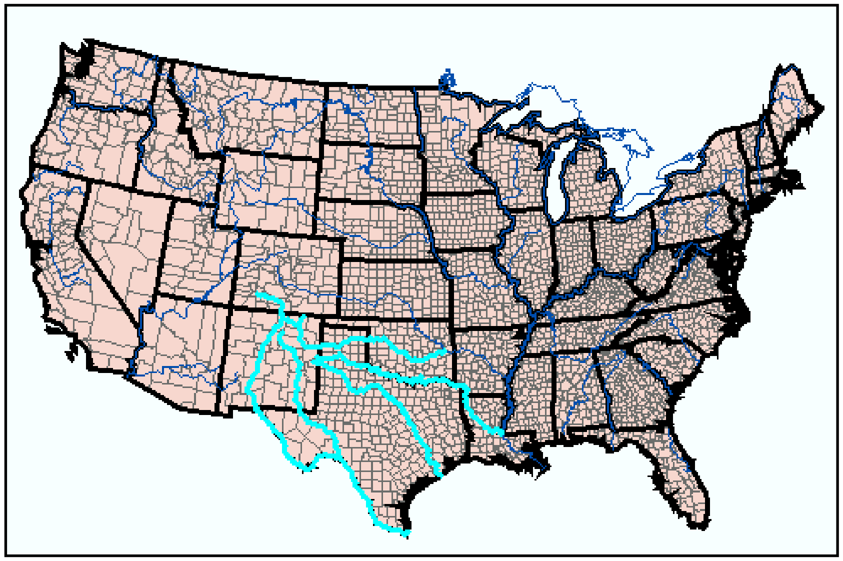


Select counties that are within 200 miles of Denver

More examples

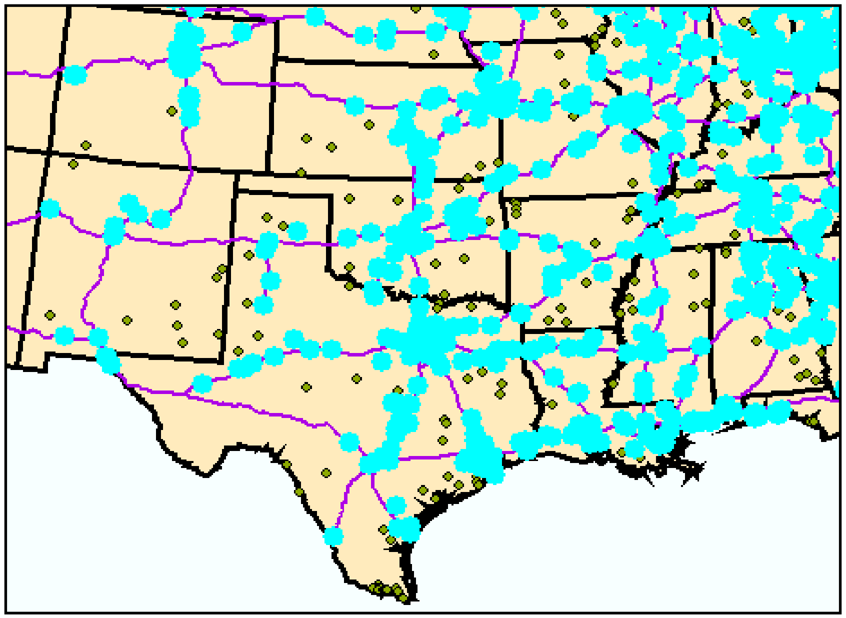


Select counties that intersect rivers

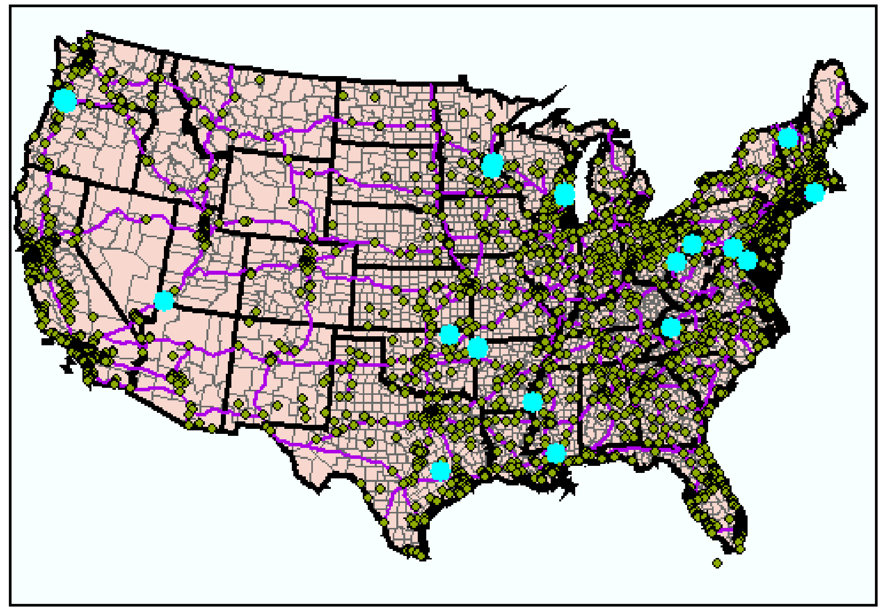


Select rivers that intersect Texas

More examples



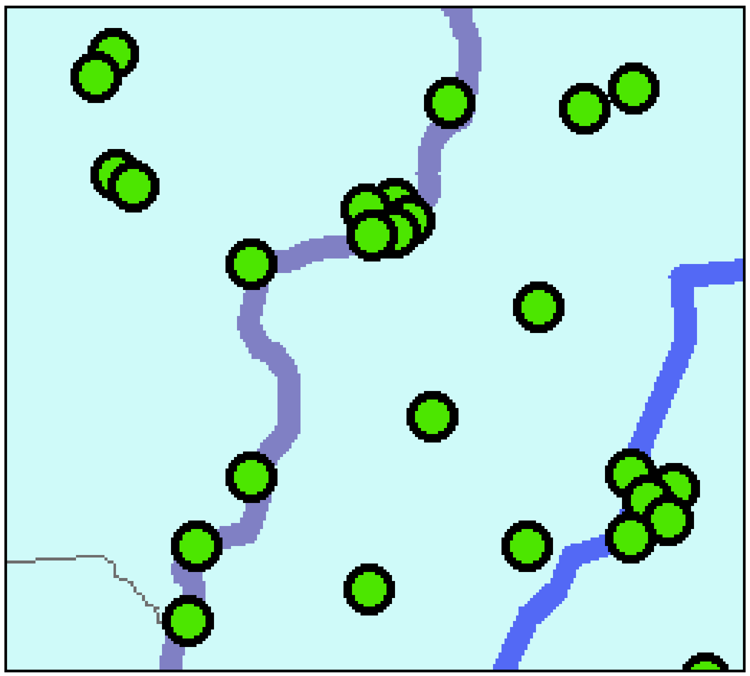
Select cities that are within 20 miles of an interstate highway



Select cities that are within counties named Washington

Scale and accuracy issues

- When testing spatial relationships, consider the **possibility** that **features are not exactly located**.

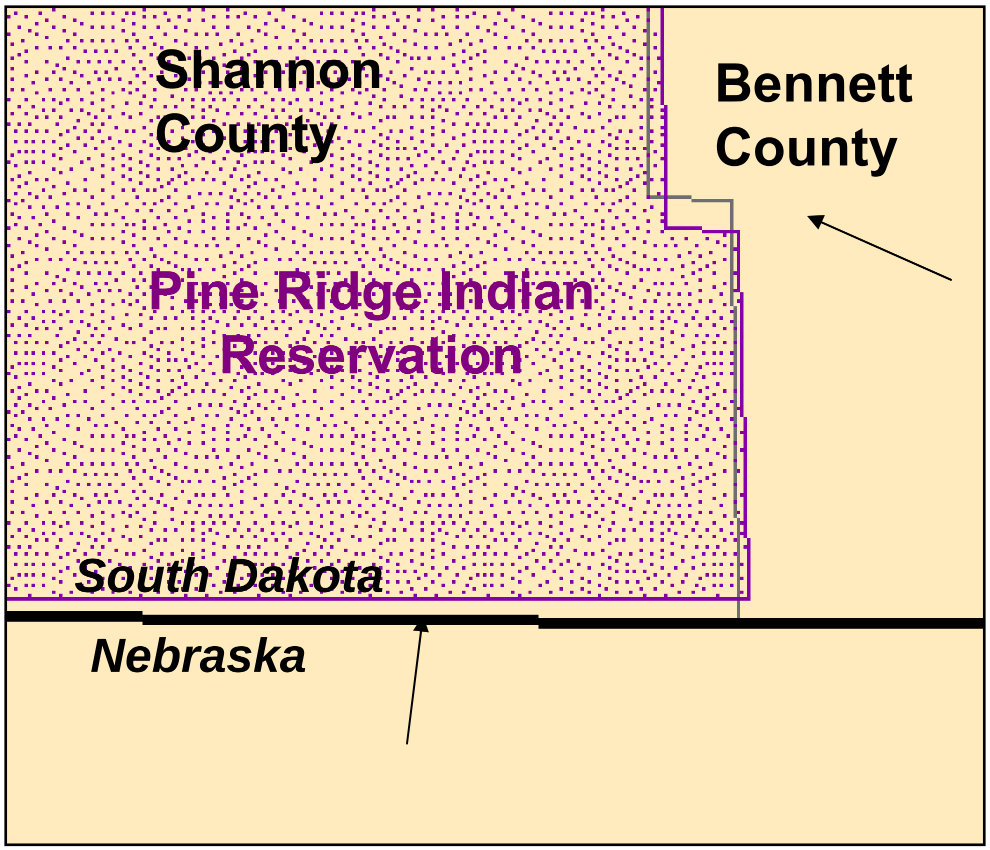


Consider **selecting cities that lie on (intersect) rivers**.

A single point or line **cannot adequately represent location at this scale**. Selection becomes a **hit or miss** affair.

One can use **buffers** to allow a little **room for error**.

Topology issues



Here, Shannon County should contain the Pine Ridge Reservation, but it does not.

- In the real world, **certain boundaries coincide**.
- This condition **won't hold true for many feature classes** unless they have been **specifically checked and corrected for logical consistency**.
- Keep in mind that your data sets may contain **topological inconsistencies** that may affect your results.

About ArcGIS

Chapter 5.

Queries

General information about queries

Queries in ArcMap

- **Interactive selection**
  - Choose features by **pointing to them on the screen**
- **Select By Attribute**
  - Select features based on **attribute criteria**
- **Select By Location**
  - Select features based on their **spatial relationships**

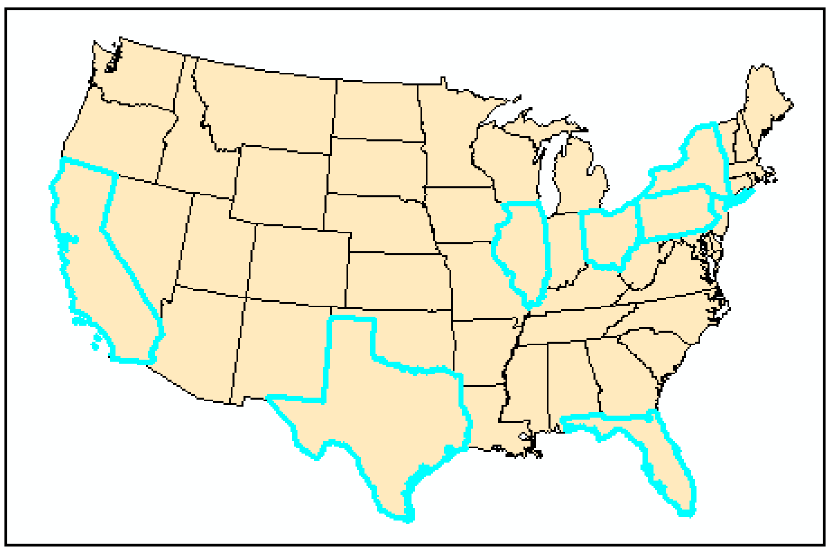
Viewing selected features

Attributes of States

FID	Shape*	AREA	STATE_NAME	STATE_FIPS	SUB_REGION	STAT
14	Polygon	8172.561	Massachusetts	25	N Eng	MA
15	Polygon	77330.258	Nebraska	31	W N Cen	NE
16	Polygon	48561.751	New York	36	Mid Atl	NY
17	Polygon	45360.118	Pennsylvania	42	Mid Atl	PA
18	Polygon	4976.566	Connecticut	09	N Eng	CT
19	Polygon	1044.881	Rhode Island	44	N Eng	RI

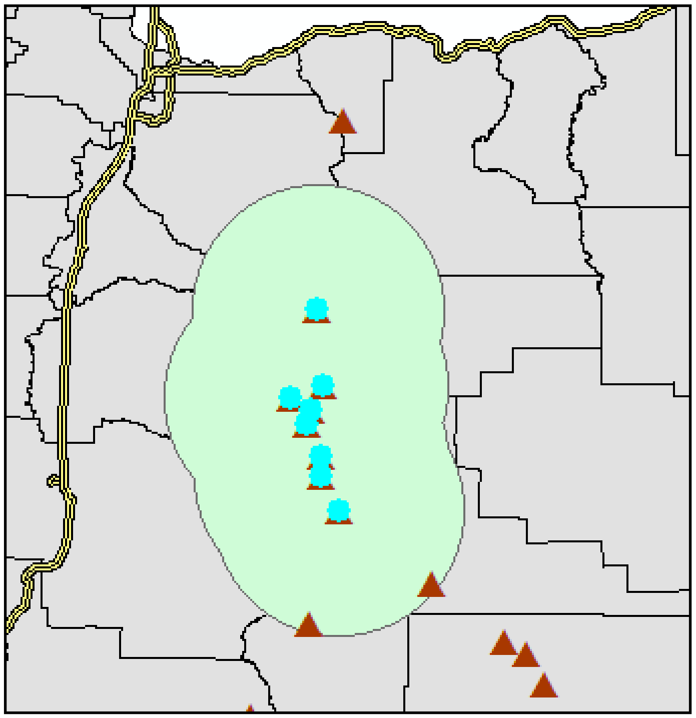
Record: 1 Show: All Selected Records (7 out of 51 Selected.)

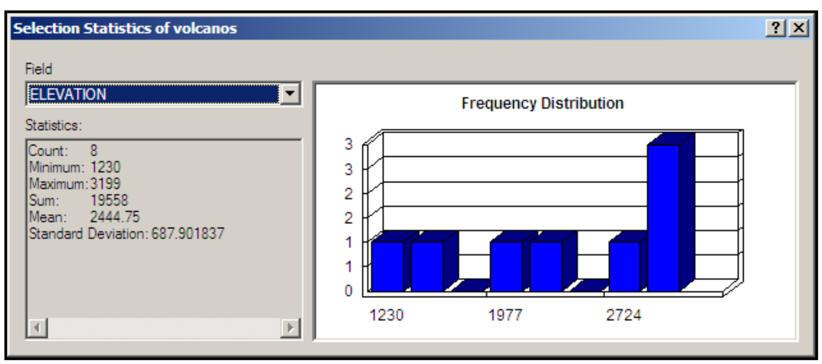
States for which POP2000>2 million



Highlighted in table
Highlighted in map

Using Selected features





- Once a **layer has a query placed upon it**, all **subsequent operations** on that layer use **ONLY the selected features**.

Volcanoes selected, then buffered

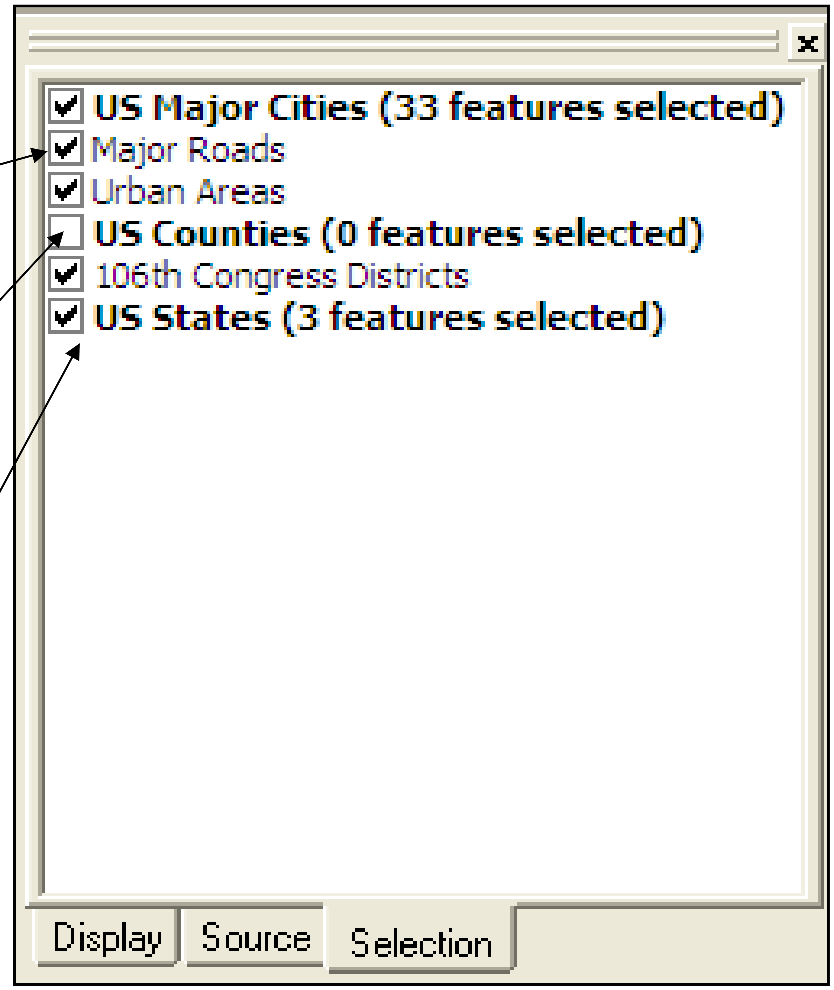
Buffer uses only selected volcanoes

Statistics only include selected volcanoes

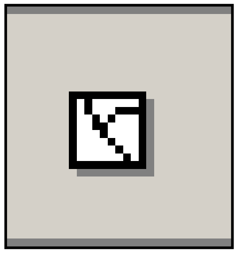
Selection states

Example: Exporting features to a new shapefile

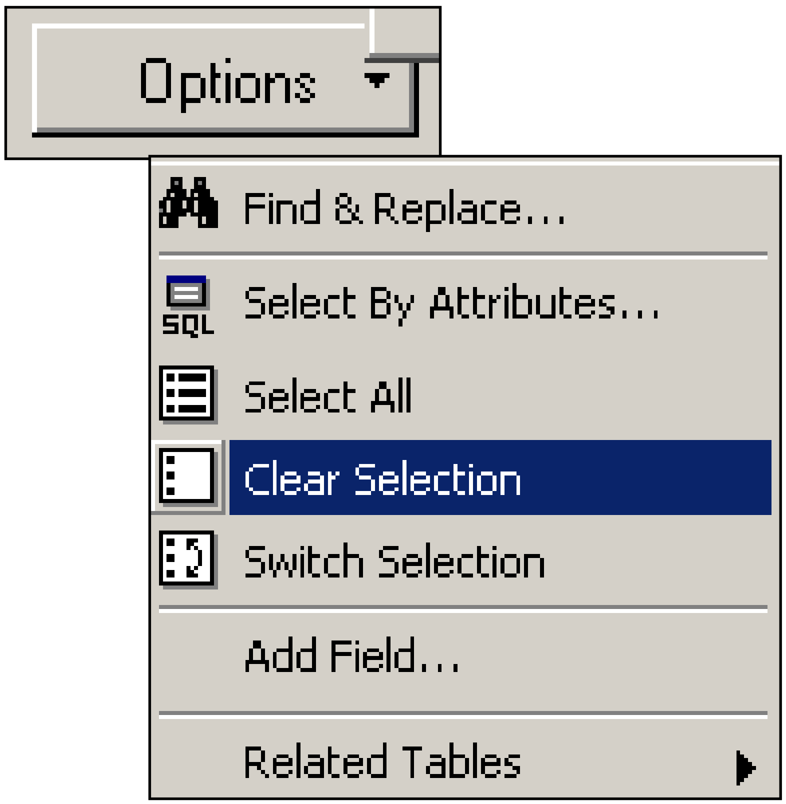
- **No selection**
  - **All** features exported
- **No selected features**
  - **No** features exported
- **Selected features**
  - **Only** selected features exported



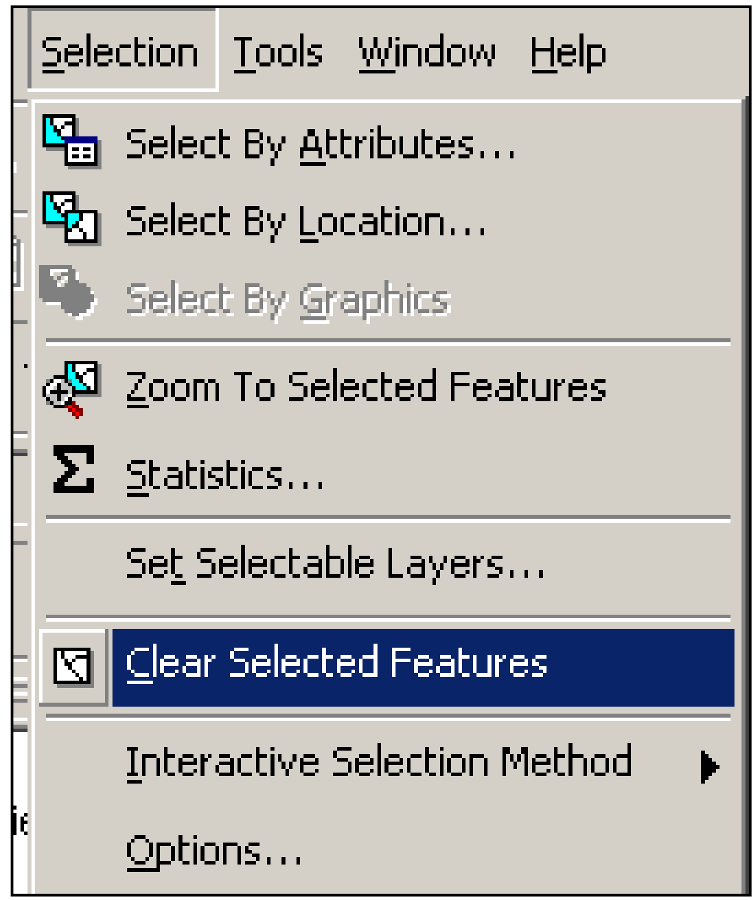
Clear Selection



On toolbar



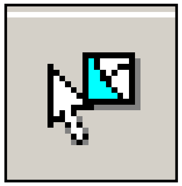
From table options menu



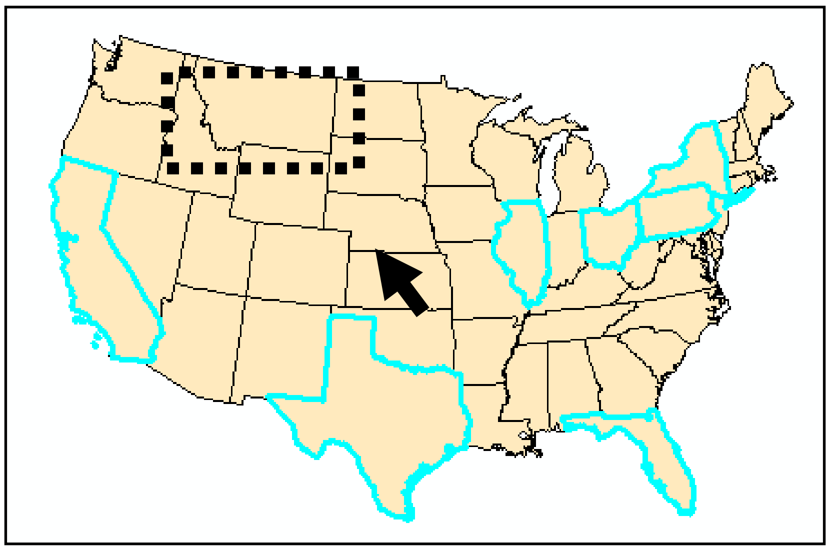
From main menu

Interactive selection

Interactive Selection



Select Features tool



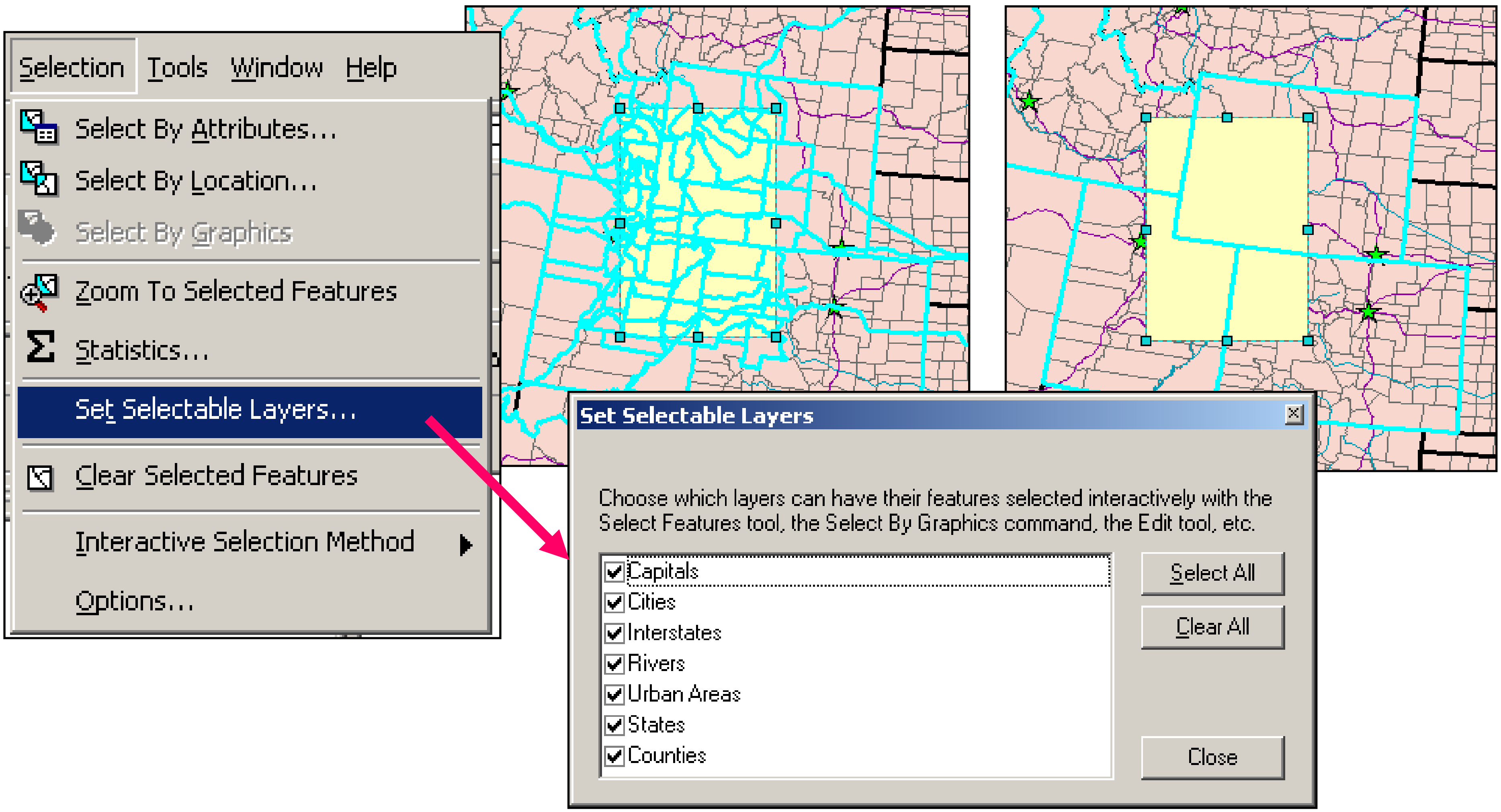
Click on feature to select

Hold down shift key to select **more than one** feature

Draw a rectangle that **passes through features** to be selected.

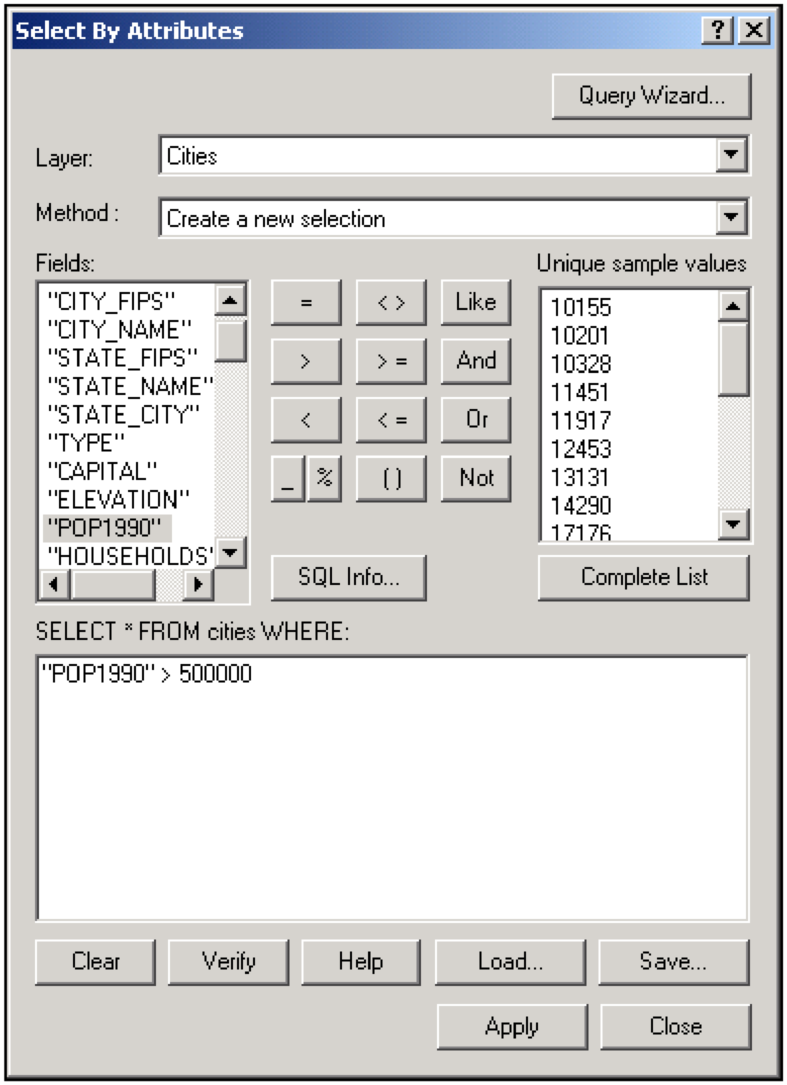
Selectable Layers

All layers selectable **States selectable**



Select by Attributes
Select by Location

Select By Attributes



Some Valid Queries

SELECT *FROM cities WHERE "POP1990" >= 500000

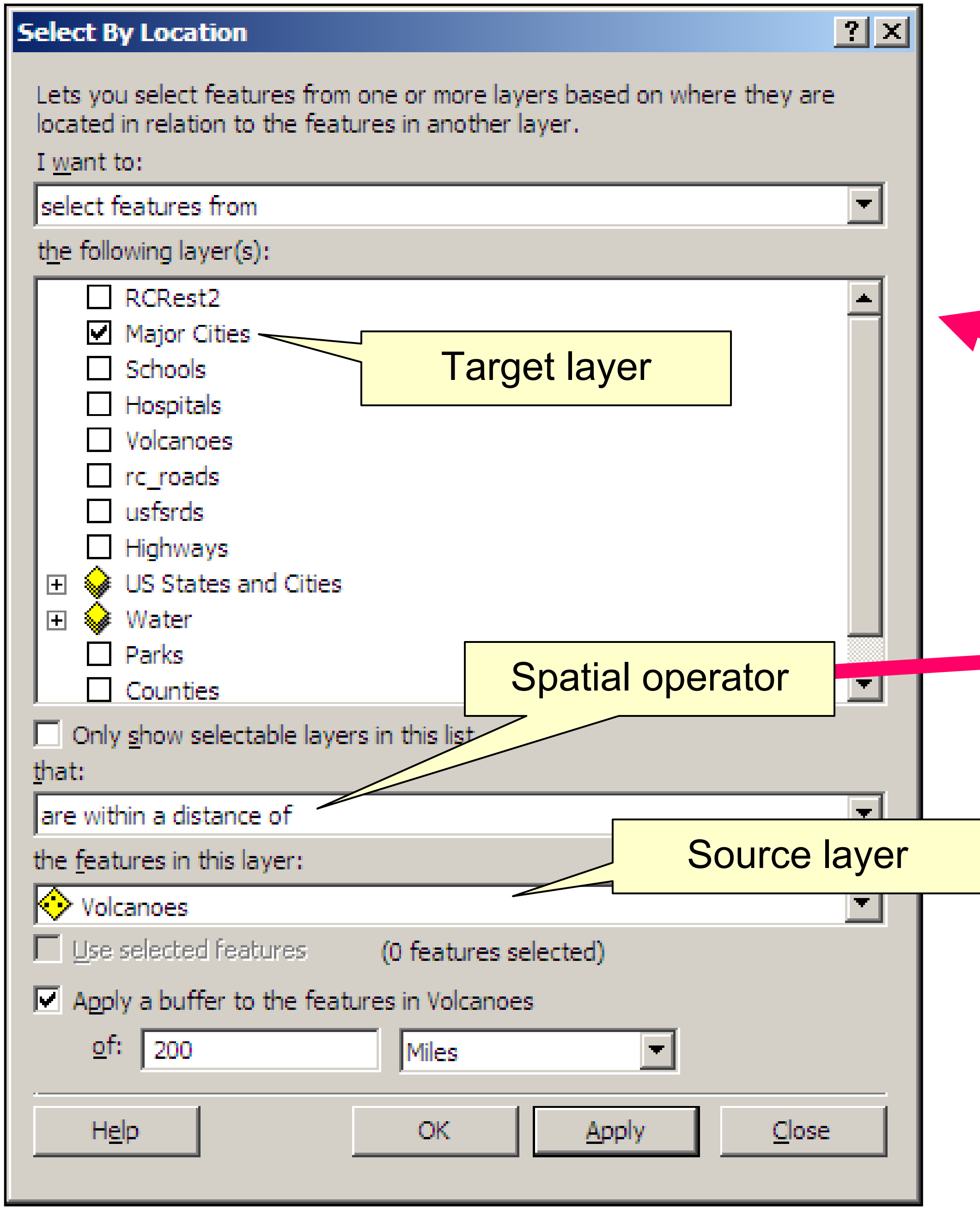
SELECT *FROM counties WHERE “BEEFCOW_92” < “BEEFCOW_87”

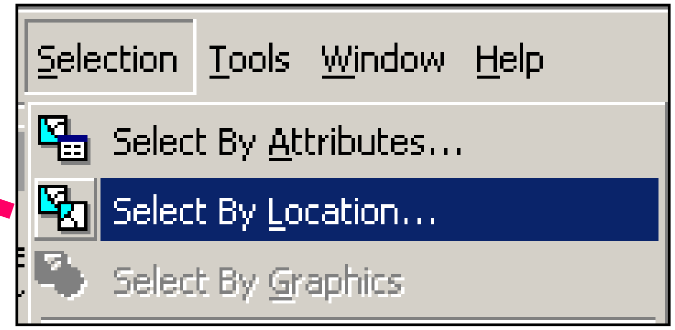
SELECT *FROM parcels WHERE “LU-CODE” = 42 AND “VALUE” > 50000

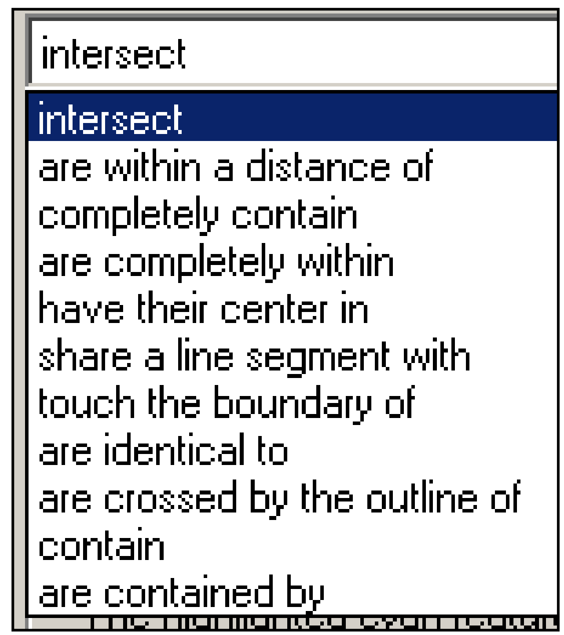
SELECT *FROM rentals WHERE “RENT” > 700 AND “RENT” < 1500

Note: Shapefile tables use quotes for field names; geodatabase tables use brackets

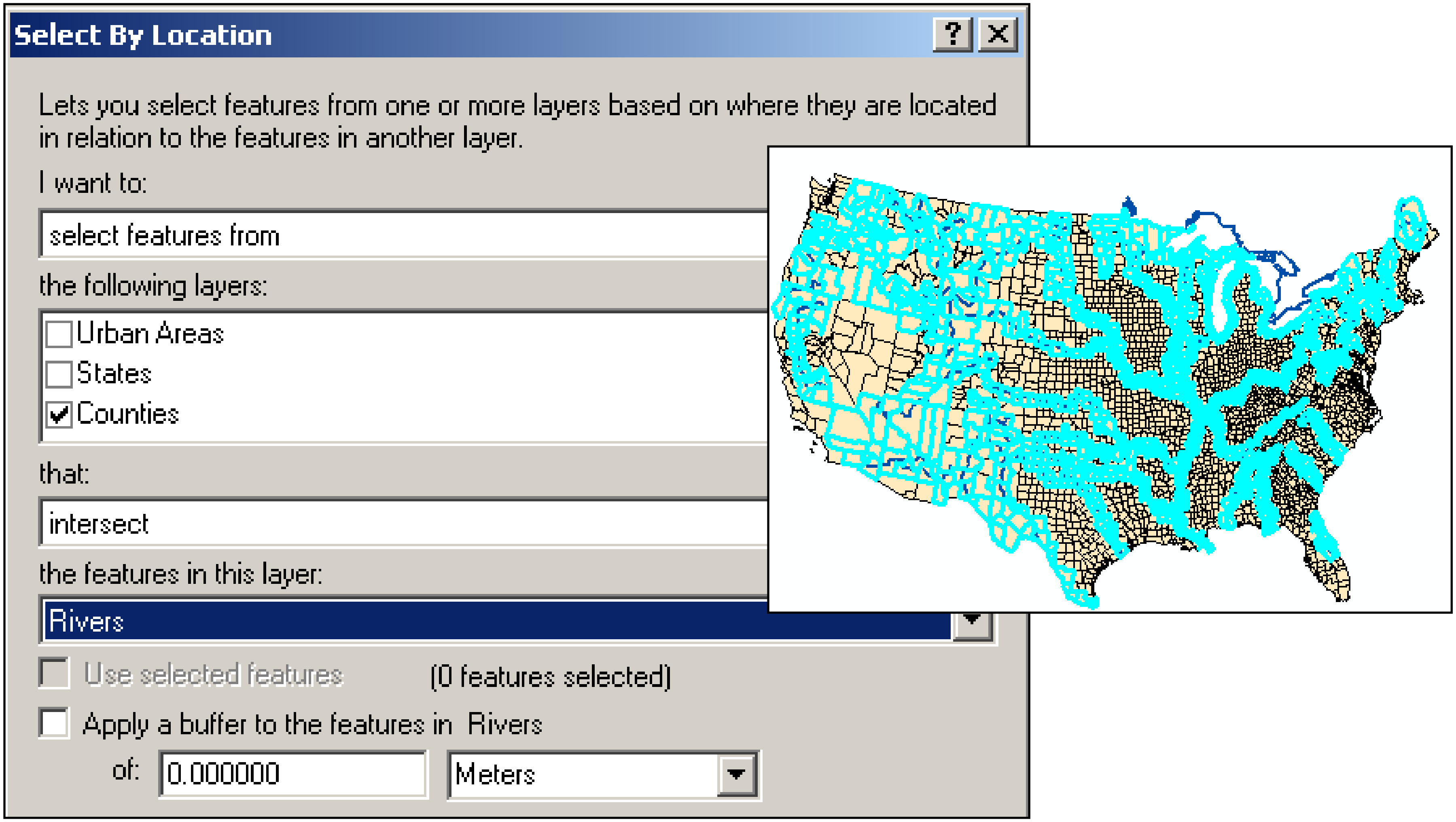
Select by Location



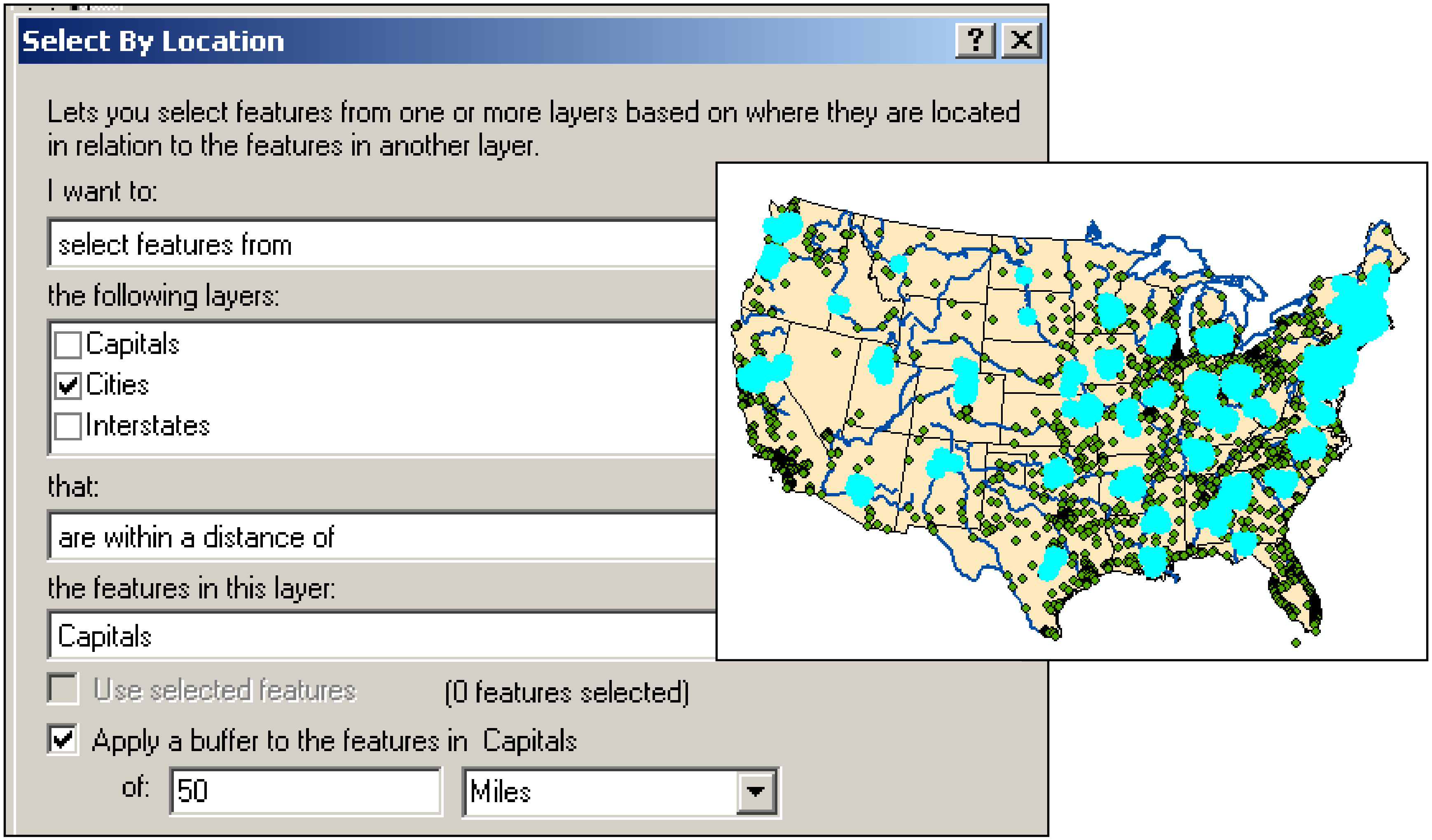




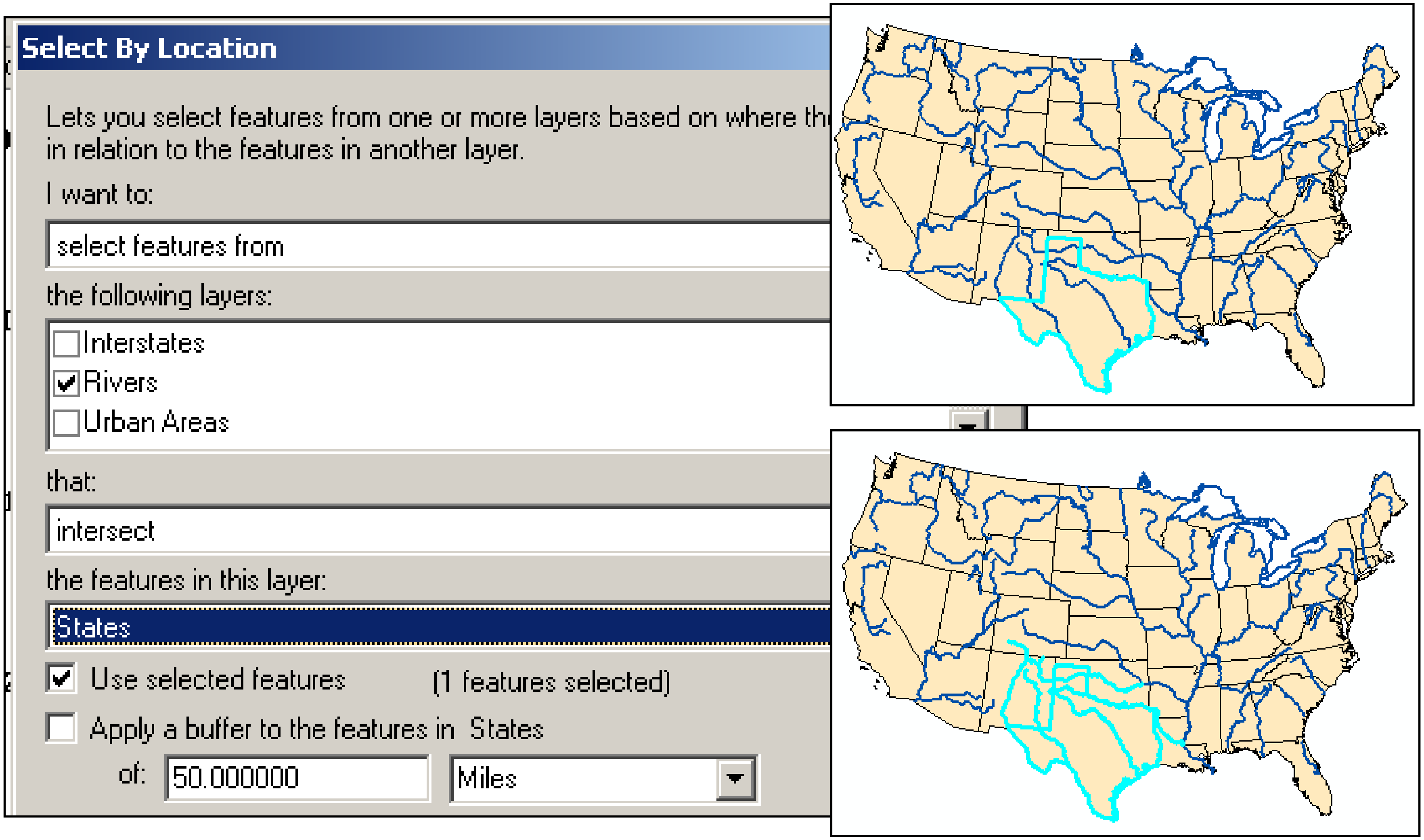
Intersect


Select By Location
Lets you select features from one or more layers based on where they are located in relation to the features in another layer.
I want to:
select features from
the following layers:
Urban Areas
States
Counties
that:
intersect
the features in this layer:
Rivers
Use selected features
(0 features selected)
Apply a buffer to the features in Rivers
of:
0.000000
Meters

Within distance of

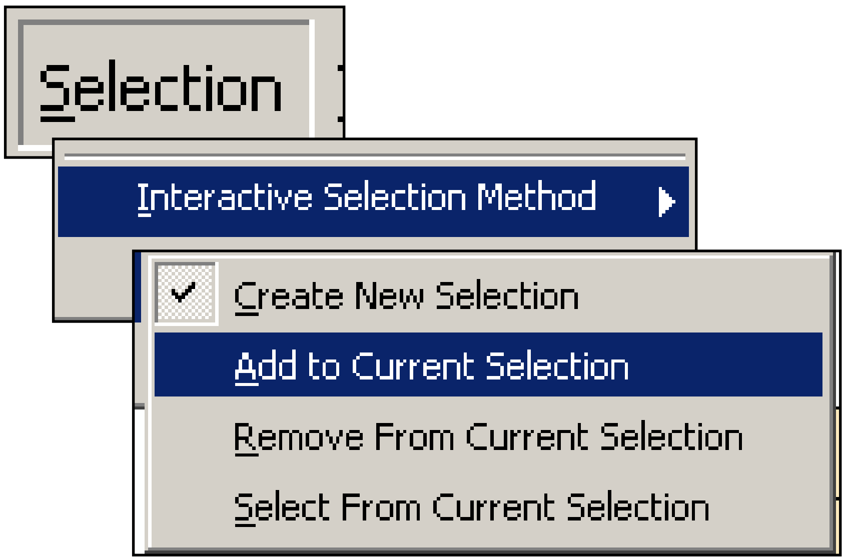


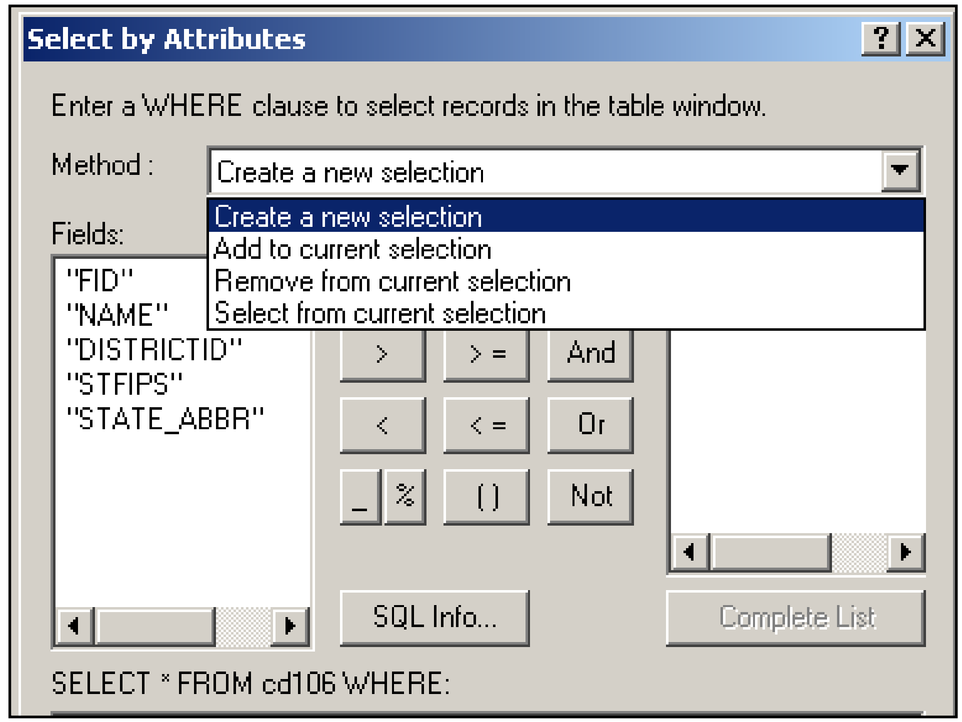
Using a selected set



Selection methods

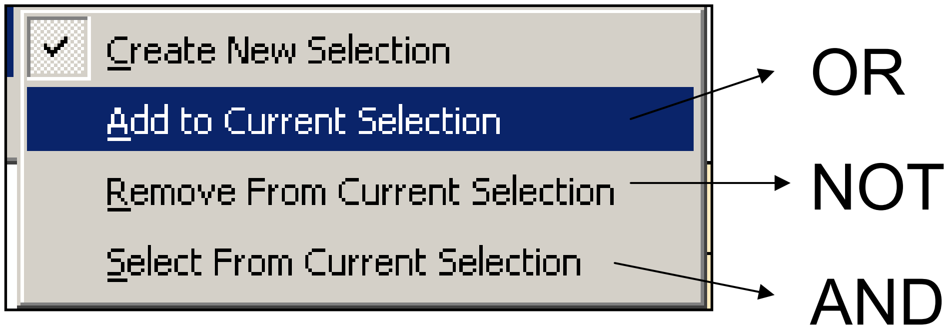
Selection methods



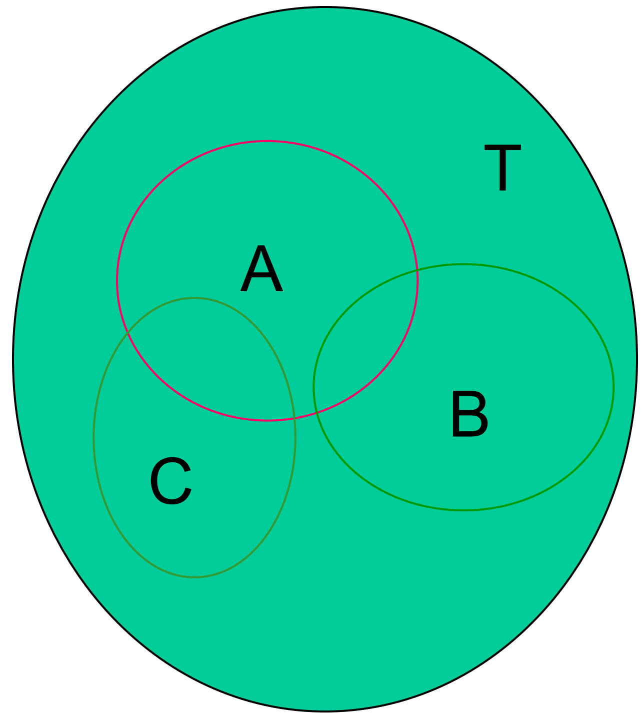


Available for all three types of selection

The Boolean Two-Step



Applying selection methods facilitates using multiple steps to apply multiple criteria—like using Boolean operators.

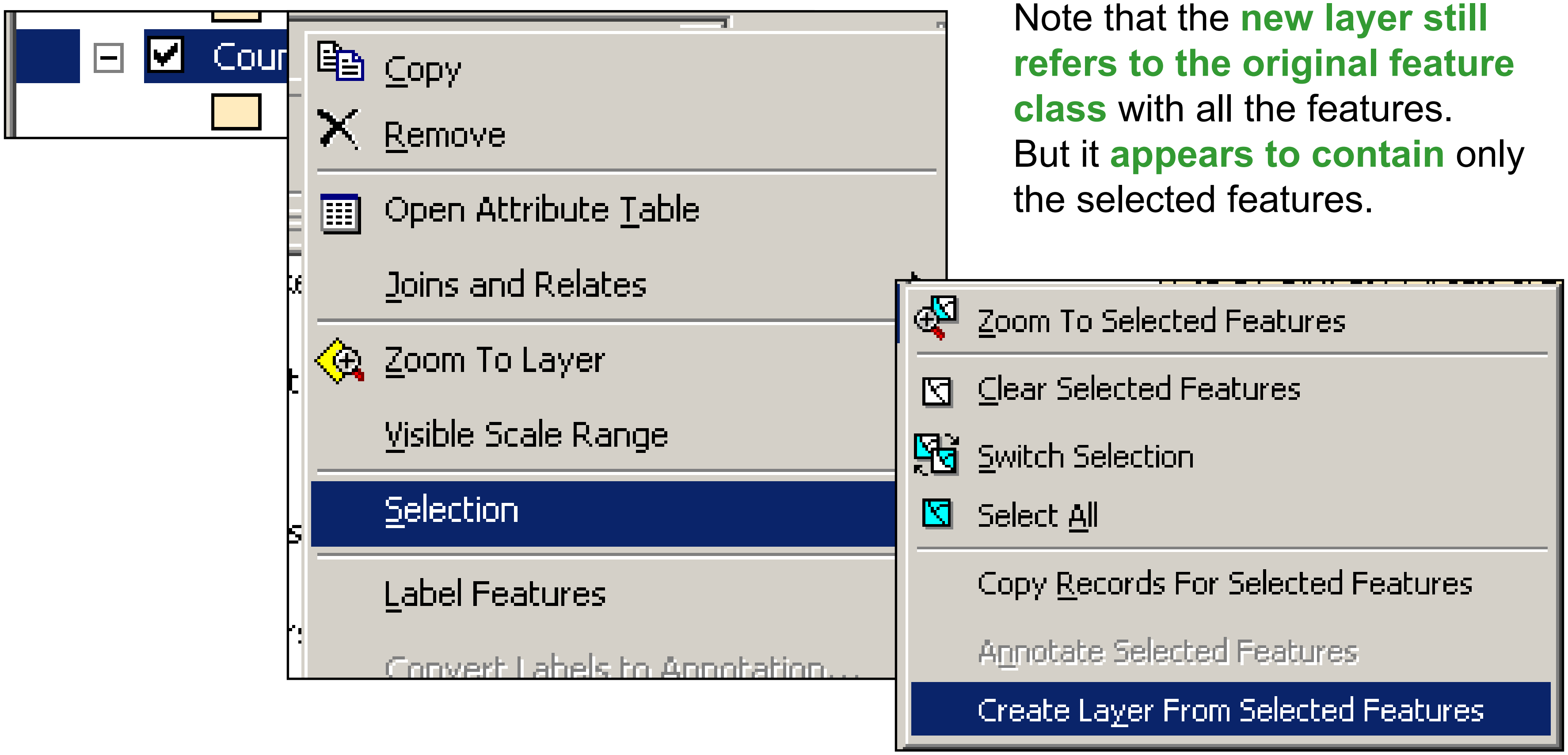


A OR B	Create new selection A; Add B to current selection
A AND B	Create new selection A; Select B from current selection
A NOT B	Create new selection A; Remove B from current selection

Creating layers from queries

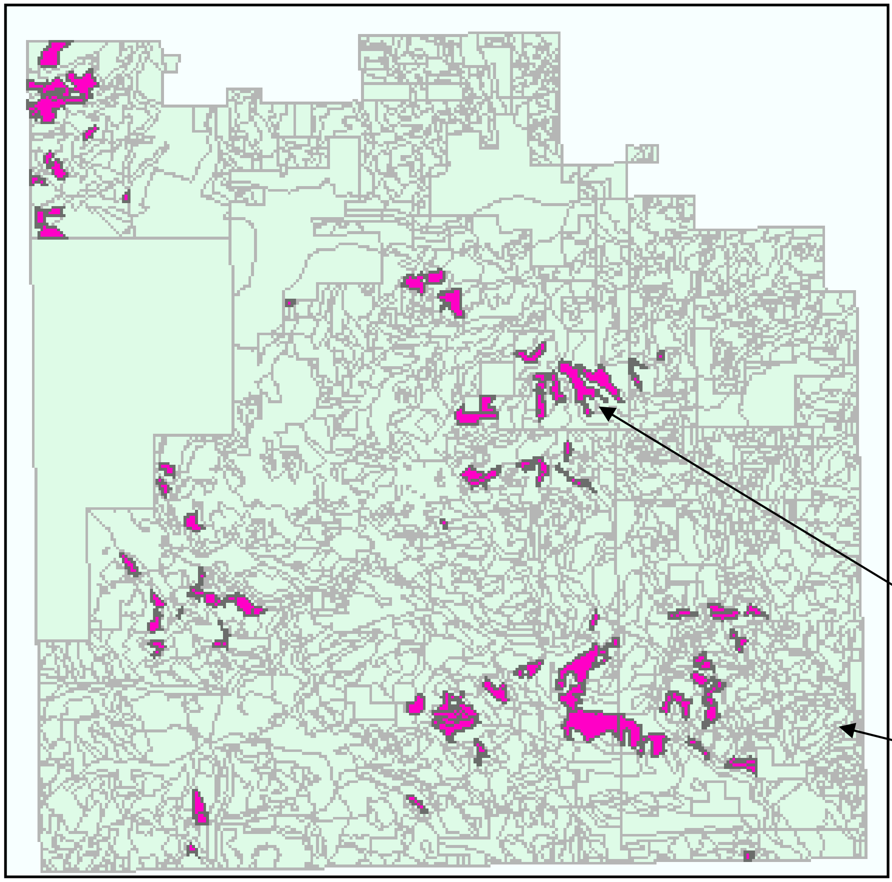
Creating layers

- **Common operation** following a query
- Creates a **new layer** with **only the selected features**



Note that the **new layer still refers to the original feature class** with all the features. But it **appears to contain** only the selected features.

Layers based on selections

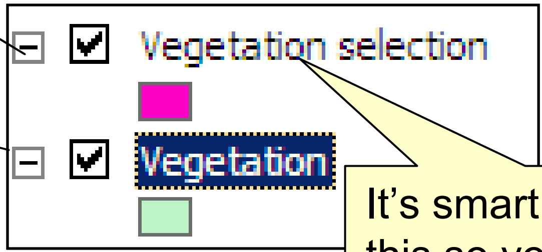


[cover_type] = ‘Aspen’

Still **based on one original file** shared by both layers

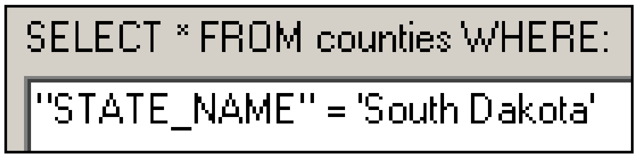
Shows only a **selected subset** in the map and in the table

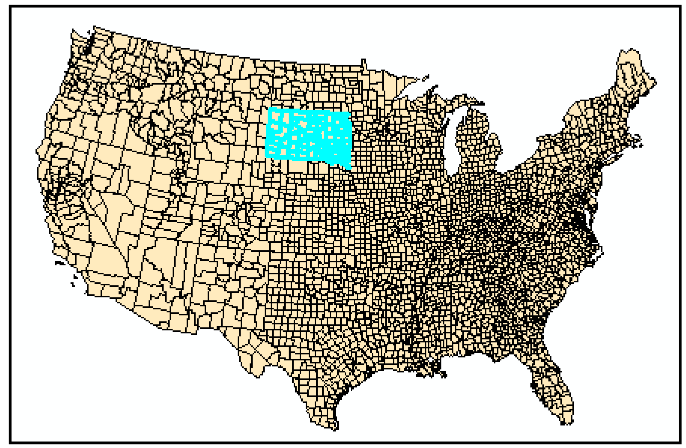
Use as **input for a tool**, e.g. buffer only the aspen stands

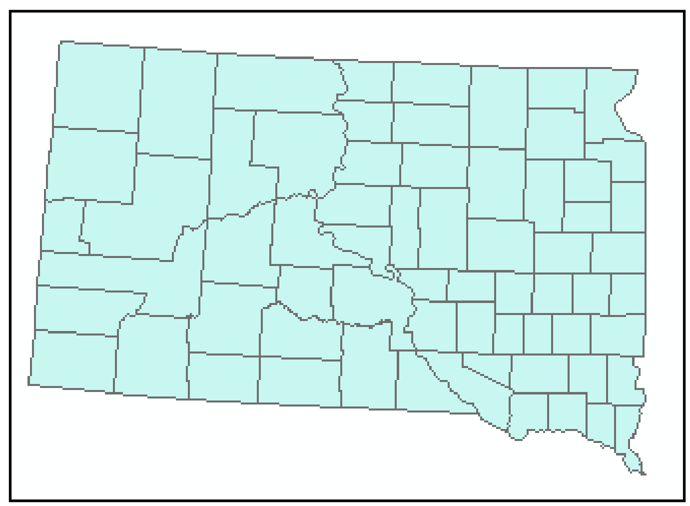


It’s smart to rename this so you can remember what it is.

Creating layers







Next Topic:

Spatial Joins