Spatial Analysis and Modeling
(GIST 4302/5302)

Guofeng Cao
Department of Geosciences
Texas Tech University
Outline

• Last week, we learned:
  – Review of map projection
  – Characteristics of spatial data
  – Types of spatial data

• This week, we will learn:
  – Concepts of database
  – Representation of different types of spatial data in GIS
  – Commonly used spatial operators in GIS
Database Fundamentals
A computer system can be thought of as comprising four major subsystems:

- Processing
- Storage
- Control
- Input/output
Review: Bits and Bytes

• Data stored in a computer system is measured in **bits**
  – each *bit* records one of two possible states
    • 0 (off, false)
    • 1 (on, true)
  – *Bits* are amalgamated into **bytes (8 bits)**
    • Each *byte* represents a single character
    • A character may be encoded using 7 *bits* with an extra *bit* used as a sign of positive or negative
      – **Question**: Can I use byte to represent the elevation of the Everest in meters?
    • Megabytes ($2^{20}$ bytes)
Database

• A **database** is a collection of data organized in such a way that a computer can efficiently store and retrieve data
  – A repository of data that is logically related

• A database is created and maintained using a general-purpose piece of software called a database management system (**DBMS**)
The Database Approach

• Before databases, computers were primarily used to convert data between different formats
  – “The computer as a giant calculator”
• Databases treat computers as useful repositories of data
  – “The computer as data repository”
• Most applications (including GIS) require a balance of processing and storage
Databases in a Nutshell

• In order to be effective, databases must offer the following functions:
  – Reliability
  – Integrity
  – Security
  – User views
  – User interface
  – Data independence
  – Self-describing
  – Concurrency
  – Distributed capabilities
  – High performance

• All these functions are managed by the DBMS
Example of Concurrency: Lost update

- Lost update can occur when transactions are incorrectly **interleaved**

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$B$</th>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1000$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X$</td>
<td>$\leftarrow B$</td>
<td>$\leftarrow B$</td>
<td>$1000$</td>
<td>$1000$</td>
<td></td>
</tr>
<tr>
<td>$X$</td>
<td>$\leftarrow X + 300$</td>
<td>$\leftarrow Y - 400$</td>
<td>$1000$</td>
<td>$1300$</td>
<td>$1000$</td>
</tr>
<tr>
<td>$B$</td>
<td>$\leftarrow X$</td>
<td>$\leftarrow Y$</td>
<td>$1300$</td>
<td>$1300$</td>
<td>$600$</td>
</tr>
</tbody>
</table>
Common Database Applications

• Home/office database
  – Simple applications (e.g., restaurant menus)

• Commercial database
  – Store the information for businesses (e.g. customers, employees)

• Engineering database
  – Used to store engineering designs (e.g. CAD)

• Image and multimedia database
  – Store image, audio, video data

• Geodatabase/spatial database
  – Store a combination of spatial and non-spatial data
Elements of a DBMS

- Query language
- Query compiler
- Runtime database processor
- Constraint enforcer
- Stored data manager
- System catalog/data dictionary
The Relational Model

• The relational model is one of the most commonly used architecture in database
  – A relational database is a collection of relations, often just called tables
  – Each relation has a set of attributes
  – The data in the relation is structured as a set of rows, often called tuples
  – Each tuple consists of data items for each attribute
  – Each cell in a tuple contains a single value
  – A relational database management system (RDBMS) is the software that manages a relational database
An Example Relation

<table>
<thead>
<tr>
<th>TITLE</th>
<th>DIRECTOR</th>
<th>CNTRY</th>
<th>YEAR</th>
<th>LNGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Bug’s Life</td>
<td>Lasseter</td>
<td>USA</td>
<td>1998</td>
<td>96</td>
</tr>
<tr>
<td>Traffic</td>
<td>Soderbergh</td>
<td>USA</td>
<td>2000</td>
<td>147</td>
</tr>
<tr>
<td>Die Another Day</td>
<td>Tamahori</td>
<td>UK</td>
<td>2002</td>
<td>132</td>
</tr>
<tr>
<td>Malcolm X</td>
<td>Lee</td>
<td>USA</td>
<td>1992</td>
<td>194</td>
</tr>
<tr>
<td>American Beauty</td>
<td>Mendes</td>
<td>USA</td>
<td>1999</td>
<td>122</td>
</tr>
<tr>
<td>Eyes Wide Shut</td>
<td>Kubrick</td>
<td>USA</td>
<td>1999</td>
<td>159</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Relations

- A **relation** is basically a “table”
- A **relation scheme** is the set of attribute names and the **domain** (data type) for each attribute name
- A **database scheme** is a set of relation schemes
- In a relation:
  - Each tuple contains as many values as there are attributes in the relation scheme
  - Each data item is drawn from the domain for its attribute
  - The order of tuples is not significant
  - Tuples in a relation are all distinct from each other
- The **degree** of a relation is its number of columns
- The **cardinality** of a relation is the number of tuples
Relation Scheme

• A **candidate key** is an attribute or minimal set of attributes that will uniquely identify each tuple in a relation

• One candidate key is usually chose as a **primary key**

```plaintext
CINEMA (CIN_ID, NAME, MANAGER, TELNO, TOWN, GRID_REF)
SCREEN (CINEMA_ID, SCREEN_NO, CAPACITY)
FILM (TITLE, DIRECTOR, CNTRY, YEAR, LNGTH)
SHOW (CINEMA_ID, SCREEN_NO, FILM_NAME, STANDARD, LUXURY)
STAR (NAME, BIRTH_YEAR, GENDER, NTY)
CAST (FILM_STAR, FILM_TITLE, ROLE)
```
Operations on Relations

• Fundamental relational operators:
  – Union, intersection, difference, product and restrict: usual set operations, but require both operands have the same schema
  – Selection: picking certain rows
  – Projection: picking certain columns
  – Join: compositions of relations

• Together, these operations and the way they are combined is called relational algebra combined:
  – An algebra whose operands are relations or variables that represent relations
Project Operator

• The **project** operator is unary
  – It outputs a new relation that has a subset of attributes
  – Identical tuples in the output relation are coalesced

Relation Sells:

<table>
<thead>
<tr>
<th></th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Chimy’s</td>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td>Cricket’s</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Cricket’s</td>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>

Prices := PROJ_{beer,price}(Sells):

<table>
<thead>
<tr>
<th></th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>
Select Operator

• The **select** operator is unary
  – It outputs a new relation that has a subset of tuples
  – A condition specifies those tuples that are required

Relation Sells:

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s Bud</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>Chimy’s Miller</td>
<td>2.75</td>
<td></td>
</tr>
<tr>
<td>Cricket’s Bud</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>Cricket’s Miller</td>
<td>3.00</td>
<td></td>
</tr>
</tbody>
</table>

ChimyMenu := \( \text{SELECT}_{\text{bar} = \text{“Chimy’s”}}(\text{Sells}) : \)

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s Bud</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>Chimy’s Miller</td>
<td>2.75</td>
<td></td>
</tr>
</tbody>
</table>
The join operator is binary

- It outputs the combined relation where tuples agree on a specified attribute (natural join)

<table>
<thead>
<tr>
<th>Sells(bar, beer, price)</th>
<th>Bars(bar, address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s Bud 2.50</td>
<td>Chimy’s 2417 Broadway St.</td>
</tr>
<tr>
<td>Chimy’s Miller 2.75</td>
<td>Cricket’s 2412 Broadway St.</td>
</tr>
<tr>
<td>Cricket’s Bud 2.50</td>
<td></td>
</tr>
<tr>
<td>Cricket’s Coors 3.00</td>
<td></td>
</tr>
</tbody>
</table>

BarInfo := Sells JOIN Bars
Note Bars.name has become Bars.bar to make the natural join “work.”

<table>
<thead>
<tr>
<th>BarInfo(bar, beer, price, address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s Bud 2.50 2417 Broadway St.</td>
</tr>
<tr>
<td>Chimy’s Miller 2.75 2417 Broadway St.</td>
</tr>
<tr>
<td>Cricket’s Bud 2.50 2412 Broadway St.</td>
</tr>
<tr>
<td>Cricket’s Coors 3.00 2412 Broadway St.</td>
</tr>
</tbody>
</table>
Join Operator

• Join is the most time-consuming of all relational operators to compute
  – In general, relational operators may not be arbitrarily reordered (left join, right join)
  – Query optimization aims to find an efficient way of processing queries, for example reordering to produce equivalent but more efficient queries
Complex Relational Operator Example

Join relations SHOW and FILM using FILM_NAME and TITLE

Select using CINEMA_ID=1

Project TITLE, DIRECTOR, CINEMA_ID, and SCREEN_NO

<table>
<thead>
<tr>
<th>TITLE</th>
<th>DIRECTOR</th>
<th>CINEMA_ID</th>
<th>SCREEN_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>Singer</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>American Beauty</td>
<td>Mendes</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>The Hours</td>
<td>Daldry</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Training Day</td>
<td>Fuqua</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traffic</td>
<td>Soderbergh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>Singer</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TITLE</th>
<th>DIRECTOR</th>
<th>CINEMA_ID</th>
<th>SCREEN_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>Singer</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>American Beauty</td>
<td>Mendes</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>The Hours</td>
<td>Daldry</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
SQL in One Slide

• Structured Query Language
• The standard for relational database management systems (RDBMS)

• Example:

Relation Sells:

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Chimy’s</td>
<td>Miller</td>
<td>2.75</td>
</tr>
<tr>
<td>Cricket’s</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Cricket’s</td>
<td>Miller</td>
<td>3.00</td>
</tr>
</tbody>
</table>

ChimyMenu := SELECT_{bar="Chimy’s"}(Sells):

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chimy’s</td>
<td>Bud</td>
<td>2.50</td>
</tr>
<tr>
<td>Chimy’s</td>
<td>Miller</td>
<td>2.75</td>
</tr>
</tbody>
</table>

• Select * from Sells where bar=‘Chimy’s’
• Select * from Sells where bar=‘Chimy’s’ orderby price asc
<table>
<thead>
<tr>
<th>FID</th>
<th>Shape</th>
<th>STATION</th>
<th>MAXDAY</th>
<th>AV8TOP</th>
<th>MONITOR</th>
<th>LAT</th>
<th>LON</th>
<th>XCOORD</th>
<th>YCOORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Point</td>
<td>60</td>
<td>16</td>
<td>7.225806</td>
<td>70060</td>
<td>34.135833</td>
<td>-117.923611</td>
<td>-117.923611</td>
<td>34.135833</td>
</tr>
<tr>
<td>1</td>
<td>Point</td>
<td>69</td>
<td>15</td>
<td>5.899194</td>
<td>70069</td>
<td>34.176111</td>
<td>-118.315278</td>
<td>-118.315278</td>
<td>34.176111</td>
</tr>
<tr>
<td>2</td>
<td>Point</td>
<td>72</td>
<td>14</td>
<td>4.052885</td>
<td>70072</td>
<td>33.823611</td>
<td>-118.1875</td>
<td>-118.1875</td>
<td>33.823611</td>
</tr>
<tr>
<td>3</td>
<td>Point</td>
<td>74</td>
<td>13</td>
<td>7.181452</td>
<td>70074</td>
<td>34.199444</td>
<td>-118.534722</td>
<td>-118.534722</td>
<td>34.199444</td>
</tr>
<tr>
<td>4</td>
<td>Point</td>
<td>75</td>
<td>12</td>
<td>6.076613</td>
<td>70075</td>
<td>34.069894</td>
<td>-117.751389</td>
<td>-117.751389</td>
<td>34.069894</td>
</tr>
<tr>
<td>5</td>
<td>Point</td>
<td>84</td>
<td>11</td>
<td>3.157258</td>
<td>70084</td>
<td>33.929167</td>
<td>-117.209722</td>
<td>-117.209722</td>
<td>33.929167</td>
</tr>
<tr>
<td>6</td>
<td>Point</td>
<td>85</td>
<td>10</td>
<td>5.201613</td>
<td>70085</td>
<td>34.016518</td>
<td>-118.059972</td>
<td>-118.059972</td>
<td>34.016518</td>
</tr>
<tr>
<td>7</td>
<td>Point</td>
<td>87</td>
<td>9</td>
<td>4.717742</td>
<td>70087</td>
<td>34.067222</td>
<td>-118.226389</td>
<td>-118.226389</td>
<td>34.067222</td>
</tr>
<tr>
<td>8</td>
<td>Point</td>
<td>88</td>
<td>8</td>
<td>6.532258</td>
<td>70088</td>
<td>34.083333</td>
<td>-118.106944</td>
<td>-118.106944</td>
<td>34.083333</td>
</tr>
<tr>
<td>9</td>
<td>Point</td>
<td>89</td>
<td>7</td>
<td>7.540323</td>
<td>70089</td>
<td>34.387542</td>
<td>-118.534722</td>
<td>-118.534722</td>
<td>34.387542</td>
</tr>
<tr>
<td>10</td>
<td>Point</td>
<td>91</td>
<td>6</td>
<td>4.891129</td>
<td>70091</td>
<td>34.050583</td>
<td>-118.454178</td>
<td>-118.454178</td>
<td>34.050583</td>
</tr>
<tr>
<td>11</td>
<td>Point</td>
<td>94</td>
<td>5</td>
<td>4.149194</td>
<td>70094</td>
<td>33.923577</td>
<td>-118.370856</td>
<td>-118.370856</td>
<td>33.923577</td>
</tr>
<tr>
<td>12</td>
<td>Point</td>
<td>96</td>
<td>4</td>
<td>7.266129</td>
<td>70096</td>
<td>34.690122</td>
<td>-118.133444</td>
<td>-118.133444</td>
<td>34.690122</td>
</tr>
<tr>
<td>13</td>
<td>Point</td>
<td>3176</td>
<td>3</td>
<td>4.100806</td>
<td>30176</td>
<td>33.820278</td>
<td>-117.9125</td>
<td>-117.9125</td>
<td>33.820278</td>
</tr>
<tr>
<td>14</td>
<td>Point</td>
<td>3177</td>
<td>2</td>
<td>4.737903</td>
<td>30177</td>
<td>33.926111</td>
<td>-117.951389</td>
<td>-117.951389</td>
<td>33.926111</td>
</tr>
<tr>
<td>15</td>
<td>Point</td>
<td>3186</td>
<td>1</td>
<td>4.600806</td>
<td>30186</td>
<td>33.6275</td>
<td>-117.690278</td>
<td>-117.690278</td>
<td>33.6275</td>
</tr>
<tr>
<td>16</td>
<td>Point</td>
<td>3195</td>
<td>0</td>
<td>4.358871</td>
<td>30195</td>
<td>33.675111</td>
<td>-117.926963</td>
<td>-117.926963</td>
<td>33.675111</td>
</tr>
<tr>
<td>17</td>
<td>Point</td>
<td>4137</td>
<td>15</td>
<td>8.241935</td>
<td>33137</td>
<td>33.85</td>
<td>-116.543056</td>
<td>-116.543056</td>
<td>33.85</td>
</tr>
<tr>
<td>18</td>
<td>Point</td>
<td>4144</td>
<td>14</td>
<td>8.899194</td>
<td>33144</td>
<td>34.010278</td>
<td>-117.426389</td>
<td>-117.426389</td>
<td>34.010278</td>
</tr>
<tr>
<td>19</td>
<td>Point</td>
<td>4149</td>
<td>13</td>
<td>8.899194</td>
<td>33149</td>
<td>33.708333</td>
<td>-117.243056</td>
<td>-117.243056</td>
<td>33.708333</td>
</tr>
<tr>
<td>20</td>
<td>Point</td>
<td>4150</td>
<td>12</td>
<td>7.16129</td>
<td>33150</td>
<td>33.925</td>
<td>-116.876389</td>
<td>-116.876389</td>
<td>33.925</td>
</tr>
<tr>
<td>21</td>
<td>Point</td>
<td>4157</td>
<td>11</td>
<td>8.133065</td>
<td>33157</td>
<td>33.714555</td>
<td>-116.233894</td>
<td>-116.233894</td>
<td>33.714555</td>
</tr>
<tr>
<td>22</td>
<td>Point</td>
<td>4158</td>
<td>10</td>
<td>7.858771</td>
<td>33158</td>
<td>33.674018</td>
<td>-117.321144</td>
<td>-117.321144</td>
<td>33.674018</td>
</tr>
<tr>
<td>23</td>
<td>Point</td>
<td>4162</td>
<td>9</td>
<td>7.962097</td>
<td>33162</td>
<td>33.975622</td>
<td>-117.332633</td>
<td>-117.332633</td>
<td>33.975622</td>
</tr>
<tr>
<td>24</td>
<td>Point</td>
<td>5175</td>
<td>8</td>
<td>8.149194</td>
<td>33175</td>
<td>34.103611</td>
<td>-116.629178</td>
<td>-116.629178</td>
<td>34.103611</td>
</tr>
<tr>
<td>25</td>
<td>Point</td>
<td>5181</td>
<td>7</td>
<td>11.649194</td>
<td>33181</td>
<td>34.243889</td>
<td>-117.273611</td>
<td>-117.273611</td>
<td>34.243889</td>
</tr>
<tr>
<td>26</td>
<td>Point</td>
<td>5197</td>
<td>6</td>
<td>8.604839</td>
<td>33197</td>
<td>34.094444</td>
<td>-117.504178</td>
<td>-117.504178</td>
<td>34.094444</td>
</tr>
<tr>
<td>27</td>
<td>Point</td>
<td>5203</td>
<td>5</td>
<td>9.947581</td>
<td>33203</td>
<td>34.107222</td>
<td>-117.273611</td>
<td>-117.273611</td>
<td>34.107222</td>
</tr>
<tr>
<td>28</td>
<td>Point</td>
<td>5204</td>
<td>4</td>
<td>10.274194</td>
<td>33204</td>
<td>34.066667</td>
<td>-117.151389</td>
<td>-117.151389</td>
<td>34.066667</td>
</tr>
<tr>
<td>29</td>
<td>Point</td>
<td>5212</td>
<td>3</td>
<td>8.524194</td>
<td>33212</td>
<td>33.984644</td>
<td>-117.151389</td>
<td>-117.151389</td>
<td>33.984644</td>
</tr>
<tr>
<td>30</td>
<td>Point</td>
<td>5213</td>
<td>2</td>
<td>4.447917</td>
<td>33213</td>
<td>34.262272</td>
<td>-117.188543</td>
<td>-117.188543</td>
<td>34.262272</td>
</tr>
<tr>
<td>31</td>
<td>Point</td>
<td>591</td>
<td>1</td>
<td>8.241935</td>
<td>35091</td>
<td>34.143889</td>
<td>-117.851389</td>
<td>-117.851389</td>
<td>34.143889</td>
</tr>
</tbody>
</table>

**Select By Attributes**

Layer: oz96

Method: Create a new selection

```
SELECT * FROM oz96 WHERE
"MAXDAY" >= 9
```
Relational Databases and Spatial data

• Several issues prevent unmodified databases being useful for spatial data
  – Structure of spatial data does not naturally fit with tables
  – Performance is impaired by the need to perform multiple joins with spatial data
  – Indexes are non-spatial in a conventional relational database

• An **extensible RDBMS** offers some solutions to these problems with
  – user defined data types
  – user-defined operations
  – user-defined indexes and access methods
  – active database functions (e.g., triggers)
Representation of Spatial Data
Representation of Spatial Data Models

- **Object-based model:** treats the space as populated by discrete, identifiable entities each with a geospatial reference
  - Buildings or roads fit into this view
  - GIS Softwares: ArcGIS

- **Field-based model:** treats geographic information as collections of spatial distributions
  - Distribution may be formalized as a mathematical function from a spatial framework to an attribute domain
  - Patterns of topographic altitudes, rainfall, and temperature fit neatly into this view.
  - GIS Software: Grass
Relational Models

- Tuples recording annual weather conditions at different locations

<table>
<thead>
<tr>
<th>Location</th>
<th>AvTemp</th>
<th>HiTemp</th>
<th>LoTemp</th>
<th>Precip</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>11</td>
<td>35</td>
<td>11</td>
<td>591</td>
</tr>
<tr>
<td>New York</td>
<td>12</td>
<td>41</td>
<td>26</td>
<td>1200</td>
</tr>
<tr>
<td>Cairo</td>
<td>21</td>
<td>45</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Wellington</td>
<td>12</td>
<td>30</td>
<td>2</td>
<td>1229</td>
</tr>
<tr>
<td>Reykjavik</td>
<td>4</td>
<td>23</td>
<td>16</td>
<td>817</td>
</tr>
<tr>
<td>New Delhi</td>
<td>25</td>
<td>45</td>
<td>0</td>
<td>703</td>
</tr>
<tr>
<td>Moscow</td>
<td>4</td>
<td>35</td>
<td>42</td>
<td>599</td>
</tr>
</tbody>
</table>

The field-based and object-based approaches are attempts to impose structure and pattern on such data.
Object-based Approach

- Clumps a relation as single or groups of tuples
  - Certain groups of measurements of climatic variables can be grouped together into a finite set of types

<table>
<thead>
<tr>
<th></th>
<th>AvTemp</th>
<th>HiTemp</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hot region</strong></td>
<td>21</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td><strong>Mild region</strong></td>
<td>11</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td><strong>Cold region</strong></td>
<td>4</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td><strong>Spatial reference</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cairo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Delhi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>London</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wellington</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reykjavik</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moscow</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object-based Example

SIDS Cases in North Carolina
Field-based approach

• Treats information as a collection of fields
  – Each *field* defines the spatial variation of an attribute as a function from the set of locations to an attribute domain

<table>
<thead>
<tr>
<th>Location</th>
<th>AvTemp</th>
<th>Location</th>
<th>HiTemp</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>11</td>
<td>London</td>
<td>35</td>
</tr>
<tr>
<td>New York</td>
<td>12</td>
<td>New York</td>
<td>41</td>
</tr>
<tr>
<td>Cairo</td>
<td>21</td>
<td>Cairo</td>
<td>45</td>
</tr>
<tr>
<td>Wellington</td>
<td>12</td>
<td>Wellington</td>
<td>30</td>
</tr>
<tr>
<td>Reykjavik</td>
<td>4</td>
<td>Reykjavik</td>
<td>23</td>
</tr>
<tr>
<td>New Delhi</td>
<td>25</td>
<td>New Delhi</td>
<td>45</td>
</tr>
<tr>
<td>Moscow</td>
<td>4</td>
<td>Moscow</td>
<td>35</td>
</tr>
</tbody>
</table>

...
Object-based Approach
Entity

- Object-based models decompose an information space into objects or **entities**

- An entity must be:
  - Identifiable
  - Relevant (be of interest)
  - Describable (have characteristics)

- The frame of spatial reference is provided by the entities themselves
Example: House object

Has several attributes, such as registration date, address, owner and boundary, which are themselves objects.
Spatial objects

• Spatial objects are called “spatial” because they exist inside “space”, called the embedding space

• A set of primitive objects can be specified, out of which all others in the application domain can be constructed, using an agreed set of operations

• Point-line-polygon primitives are common in existing systems
Example: GIS analysis

• For Italy’s capital city, Rome, calculate the total length of the River Tiber which lies within 2.5 km of the Colosseum
  – First we need to model the relevant parts of Rome as objects
  – Operation *length* will act on *arc*, and *intersect* will apply to form the piece of the *arc* in common with the disc
Example: GIS analysis

• A process of discretization must convert the objects to types that are computationally tractable

• A circle may be represented as a discrete polygonal area, arcs by chains of line segments, and points may be embedded in some discrete space
Primitive Objects

• **Euclidean Space**: coordinatized model of space
  – Transforms spatial properties into properties of tuples of real numbers
  – Coordinate frame consists of a fixed, distinguished point (origin) and a pair of orthogonal lines (axes), intersecting in the origin

• Point objects
• Line objects
• Polygonal objects
Points

• A point in the Cartesian plane $\mathbb{R}^2$ is associated with a unique pair of real number $a = (x, y)$ measuring distance from the origin in the $x$ and $y$ directions. It is sometimes convenient to think of the point $a$ as a vector.

• **Scalar**: Addition, subtraction, and multiplication, e.g.,
  $$(x_1, y_1) - (x_2, y_2) = (x_1 - x_2, y_1 - y_2)$$

• **Norm**: $||a|| = \sqrt{x^2 + y^2}$

• **Distance**: $|ab| = ||a-b||$

• **Angle between vectors**:
  $$\cos \alpha = \frac{x_a x_b + y_a y_b}{||a|| \cdot ||b||}$$
The *line* incident with $a$ and $b$ is defined as the point set 
\[ \{ \lambda a + (1 - \lambda) b \mid \lambda \in \mathbb{R} \} \]

The *line segment* between $a$ and $b$ is defined as the point set 
\[ \{ \lambda a + (1 - \lambda) b \mid \lambda \in [0, 1] \} \]

The *half line* radiating from $b$ and passing through $a$ is defined as the point set 
\[ \{ \lambda a + (1 - \lambda)b \mid \lambda \geq 0 \} \]
Polygonal objects

- A *polyline* in $\mathbb{R}^2$ is a finite set of line segments (called *edges*) such that each edge end-point is shared by exactly two edges, except possibly for two points, called the *extremes* of the polyline.
- If no two edges intersect at any place other than possibly at their end-points, the polyline is *simple*.
- A polyline is *closed* if it has no extreme points.
- A (*simple*) polygon in $\mathbb{R}^2$ is the area enclosed by a simple closed polyline. This polyline forms the *boundary* of the polygon. Each end-point of an edge of the polyline is called a *vertex* of the polygon.
- A *convex* polygon has every point *intervisible*
- A *star-shaped* or *semi-convex* polygon has at least one point that is intervisible
Polygons objects

- Polyline
- Simple closed polyline
- Polygon
- Convex polygon
- Star-shaped polygon
Convexity

Visibility between points $x$, $y$, and $z$
Example: Triangulation

- Every simple polygon has a triangulation. Any triangulation of a simple polygon with $n$ vertices consists of exactly $n - 2$ triangles.

- Art Gallery Problem
  - How many cameras are needed to guard a gallery and how should they be placed?
  - Upper bound $N/3$
Related: Convex Hull
Related: Voronoi Diagram
Voronoi Diagram on Road Network
John Snow, Pumps and Cholera Outbreak
Primitive GIS Operations

• in Euclidean spaces
  – Length, bearing, area
  – Distance between objects (points, lines, polygons)
  – Centroid
  – Point in polygon
  – Buffer
  – Intersection/overlay

• In topological spaces
  – Spatial relations (within, touch, cover, ...)

Distance and angle between points

• Length of a line segment can be computed as the distance between successive pairs of points

\[ |pq| = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2} \]

• The bearing, \( \theta \), of \( q \) from \( p \) is given by the unique solution in the interval \([0, 360]\) of the simultaneous equations:

\[
\cos \theta = \frac{y_q - y_p}{|pq|} \quad \sin \theta = \frac{x_q - x_p}{|pq|}
\]
Distance from point to line

- from a point to a line implies minimum distance
- For a straight line segment, distance computation depends on whether $p$ is in $\text{middle}(l)$ or $\text{end}(l)$
- For a polyline, distance to each line segment must be calculated
- A polygon calculation is as for polyline (distance to boundary of polygon)
Area

• Let $P$ be a simple polygon (no boundary self-intersections) with vertex vectors: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $(x_1, y_1) = (x_n, y_n)$. Then the area is:

\[
\text{area}(P) = \frac{1}{2} \sum_{i=1}^{n-1} x_i y_{i+1} - x_{i+1} y_i
\]

• In the case of a triangle $pqr$

\[
\text{area}(pqr) = \frac{xy_q - x_y p + x_q y_r - x_r y_q + x_r y_p - x_p y_r}{2}
\]
Area of a simple polygon

• Note that the area may be positive or negative
• In fact, $\text{area}(pqr) = -\text{area}(qpr)$
• If $p$ is to the left of $qr$ then the area is positive, if $p$ is to the right of $qr$ then the area is negative

$$\text{side}(p, q, r) = \begin{cases} 1 & \text{if } \text{area}(pqr) > 0 \quad (p \text{ is left of } qr) \\ 0 & \text{if } \text{area}(pqr) = 0 \quad (pqr \text{ are collinear}) \\ -1 & \text{if } \text{area}(pqr) < 0 \quad (p \text{ is right of }qr) \end{cases}$$
Centroid

• The **centroid** of a polygon (or **center of gravity**) of a (simple) polygonal object \( P = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \) where \((x_1, y_1) = (x_n, y_n)\) is the point at which it would balance if it were cut out of a sheet of material of uniform density:

\[
\text{centroid}_x(P) = \frac{1}{6 \cdot \text{area}(P)} \sum_{i=1}^{n-1} (x_i + x_{i+1})(x_{i+1}y_{i+1} - x_i y_i)
\]

\[
\text{centroid}_y(P) = \frac{1}{6 \cdot \text{area}(P)} \sum_{i=1}^{n-1} (y_i + y_{i+1})(x_{i+1}y_{i+1} - x_i y_i)
\]
Point in polygon

- Determining whether a point is inside a polygon is one of the most fundamental operations in a spatial database
- **Semi-line method (ray casting)**: checks for odd or even numbers of intersections of a semi-line with polygon
- **Winding method**: sums bearings from point to polygon vertices
Is Biloxi in Mississippi?
1 intersection
= Hey Y'all!

Is New Orleans in Mississippi?
2 Intersections
= Reject-shon!
Collinearity and point on segment

- Boolean operation \texttt{colinear}(a,b,c) determine whether points \(a, b\) and \(c\) lie on the same straight line

\[
\text{Colinear}(a,b,c) = \text{true} \text{ if and only if } \text{side } (a,b,c) = 0
\]

- Operation \texttt{point\_on\_segment}(p,l) returns the Boolean value \texttt{true} if \(p \in l\) (line segment \(l\) having end-points \(q\) and \(r\))

\[
\begin{align*}
\text{Determine whether } p, q, r \text{ are collinear} \\
\text{If yes, then } p \in l \text{ if and only if } p \in (\text{minimum bounding box}) \texttt{MMB}(l)
\end{align*}
\]

\[
\begin{align*}
\min(x_q, x_r) & \leq x_p \leq \max(x_q, x_r) \\
\min(y_q, y_r) & \leq y_p \leq \max(y_q, y_r)
\end{align*}
\]
Segment intersection

- Two line segments $ab$ and $cd$ can only intersect if $a$ and $b$ are on opposite sides of $cd$ and $c$ and $d$ are on opposite sides of $ab$
- Therefore two line segments intersect if the following inequalities hold

\[
\text{side}(a, b, c) \neq \text{side}(a, b, d) \\
\text{side}(c, d, a) \neq \text{side}(c, d, b)
\]
Point of intersection

- Intersecting line segments \( l \) and \( l' \) in parametric form:
  \[
  l = \{ \lambda p + (1 - \lambda)q | \lambda \in [0, 1] \} \\
  l' = \{ \lambda' p' + (1 - \lambda')q' | \lambda' \in [0, 1] \}
  \]
- Means that there exists an \( \alpha \) and \( \beta \) such that:
  \[
  \alpha p + (1 - \alpha)q = r = \beta p' + (1 - \beta)q'
  \]
- Which solving for \( \alpha \) and \( \beta \) give:
  \[
  \alpha = \frac{(x_q - x_{q'}) (y_{p'} - y_{q'}) - (x_{p'} - x_q) (y_q - y_{q'})}{(x_{p'} - x_q) (y_p - y_q) - (x_p - x_q) (y_{p'} - y_{q'})}
  \]
  \[
  \beta = \frac{(x_p - x_q) (y_{q'} - y_q) - (x_{q'} - x_q) (y_p - y_q)}{(x_{p'} - x_q) (y_p - y_q) - (x_p - x_q) (y_{p'} - y_{q'})}
  \]
Primitive GIS operations: Overlay

- Union
- Intersect
- Erase
- Identity
- Update
- Spatial Join
- Symmetrical Difference
Overlay

• Union
  – Computes a geometric union of the input features. All features and their attributes will be written to the output feature class.
Overlay

- **Intersect**
  - Computes a geometric intersection of the input features. Features or portions of features which overlap in all layers and/or feature classes will be written to the output feature class.
Overlay

• Erase
  – Creates a feature class by overlaying the Input Features with the polygons of the Erase Features. Only those portions of the input features falling outside the erase features outside boundaries are copied to the output feature class
Overlay

• Identity
  – Computes a geometric intersection of the input features and identity features. The input features or portions thereof that overlap identity features will get the attributes of those identity features
Overlay

• Update
  – Computes a geometric intersection of the Input Features and Update Features. The attributes and geometry of the input features are updated by the update features in the output feature class.
Symmetrical difference

Features or portions of features in the input and update features that do not overlap will be written to the output feature class.
Overlay

• Spatial join
  – Joins attributes from one feature to another based on the spatial relationship. The target features and the joined attributes from the join features are written to the output feature class.
Buffer

Vector buffers

points

lines

polygons

input source features

output buffer features
Topological spatial operations: spatial relationship

• Object types with an assumed underlying topology are **point, arc, loop** and **area**

• Operations:
  
  – **boundary, interior, closure** and **connected** are defined in the usual manner
  
  – **components** returns the set of maximal connected components of an area
  
  – **extremes** acts on each object of type arc and returns the pair of points of the arc that constitute its end points
  
  – **is within** provides a relationship between a point and a simple loop, returning true if the point is enclosed by the loop
Topological spatial operations for areas

- \( X \) meets \( Y \) if \( X \) and \( Y \) touch externally in a common portion of their boundaries

- \( X \) overlaps \( Y \) if \( X \) and \( Y \) impinge into each other’s interiors
Topological spatial operations for areas

- $X$ is inside $Y$ if $X$ is a subset of $Y$ and $X$, $Y$ do not share a common portion of boundary

- $X$ covers $Y$ if $Y$ is a subset of $X$ and $X$, $Y$ touch externally in a common portion of their boundaries
Topological spatial operations

• There are an infinite number of possible topological relationships that are available between objects of type cell
<table>
<thead>
<tr>
<th>Match Option (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINS CLEMENTINI</td>
</tr>
<tr>
<td>INTERSECT</td>
</tr>
<tr>
<td>INTERSECT_3D</td>
</tr>
<tr>
<td>WITHIN_A_DISTANCE</td>
</tr>
<tr>
<td>WITHIN_A_DISTANCE_3D</td>
</tr>
<tr>
<td>CONTAINS</td>
</tr>
<tr>
<td>COMPLETELY_CONTAINS</td>
</tr>
<tr>
<td>CONTAINS CLEMENTINI</td>
</tr>
<tr>
<td>WITHIN</td>
</tr>
<tr>
<td>COMPLETELY_WITHIN</td>
</tr>
<tr>
<td>WITHIN_CLEMENTINI</td>
</tr>
<tr>
<td>ARE_IDENTIAL_TO</td>
</tr>
<tr>
<td>BOUNDARY_TOUCHES</td>
</tr>
<tr>
<td>SHARE_A_LINE_SEGMENT_WITH</td>
</tr>
<tr>
<td>CROSSED_BY_THE_OUTLINE_OF</td>
</tr>
<tr>
<td>HAVE_THEIR_CENTER_IN</td>
</tr>
<tr>
<td>CLOSEST</td>
</tr>
</tbody>
</table>
Spaghetti

- *Spaghetti* data structure represents a planar configuration of points, arcs, and areas
- Geometry is represented as a set of lists of straight-line segments
Spaghetti- example

• Each polygonal area is represented by its boundary loop
• Each loop is discretized as a closed polyline
• Each polyline is represented as a list of points

A: [1, 2, 3, 4, 21, 22, 23, 26, 27, 28, 20, 19, 18, 17]
B: [4, 5, 6, 7, 8, 25, 24, 23, 22, 21]
C: [8, 9, 10, 11, 12, 13, 29, 28, 27, 26, 23, 24, 25]
D: [17, 18, 19, 20, 28, 29, 13, 14, 15, 16]
Issues

• There is **NO** explicit representation of the topological interrelationships of the configuration, such as adjacency

• Data consistence issues
  – *Silver polygons*
  – *Data redundancy*
NAA: node arc area

• Each directed arc has exactly one start and one end node.
• Each node must be the start node or end node (maybe both) of at least one directed arc.
• Each area is bounded by one or more directed arcs.
• Directed arcs may intersect only at their end nodes.
• Each directed arc has exactly one area on its right and one area on its left.
• Each area must be the left area or right area (maybe both) of at least one directed arc.
NAA: planar decomposition

<table>
<thead>
<tr>
<th>Arc</th>
<th>Begin</th>
<th>End</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
<td>1</td>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>4</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>3</td>
<td>D</td>
<td>X</td>
</tr>
<tr>
<td>e</td>
<td>5</td>
<td>1</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>f</td>
<td>4</td>
<td>5</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>g</td>
<td>6</td>
<td>2</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>h</td>
<td>5</td>
<td>6</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>i</td>
<td>3</td>
<td>6</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>
Finding what’s Inside and what’s nearby

Involves relationship between two or more layers

**Toxic gas**  **Population**

**Ancient Forest**  **Watersheds**
What counties are within 50 miles of Interstate 10?
Spatial Index

• Efficient spatial query for massive datasets, e.g., spatial grid index, quadtree, R-tree, ..

• An example of quadtree
Field-based Approach
Spatial fields

• If the spatial framework is a Euclidean plane and the attribute domain is a subset of the set of real numbers;
  – The Euclidean plane plays the role of the horizontal xy-plane
  – The *spatial field* values give the z-coordinates, or “heights” above the plane

<table>
<thead>
<tr>
<th>Regional Climate Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagine placing a square grid over a region and measuring aspects of the climate at each node of the grid. Different fields would then associate locations with values from each of the measured attribute domains.</td>
</tr>
</tbody>
</table>
Properties of the attribute domain

• The attribute domain may contain values which are commonly classified into four levels of measurement
  – **Nominal attribute**: simple labels; qualitative; cannot be ordered; and arithmetic operators are not permissible
  – **Ordinal attribute**: ordered labels; qualitative; and cannot be subjected to arithmetic operators, apart from ordering
  – **Interval attributes**: quantities on a scale without any fixed point; can be compared for size, with the magnitude of the difference being meaningful; the ratio of two interval attributes values is not meaningful
  – **Ratio attributes**: quantities on a scale with respect to a fixed point; can support a wide range of arithmetical operations, including addition, subtraction, multiplication, and division
Continuous and differentiable fields

- **Continuous** field: small changes in location leads to small changes in the corresponding attribute value
- **Differentiable** field: rate of change (slope) is defined everywhere
- Spatial framework and attribute domain must be continuous for both these types of fields
- Every differentiable field must also be continuous, but not every continuous field is differentiable
One dimensional examples

• Fields may be plotted as a graph of attribute value against spatial framework
One dimensional examples

The field is continuous (the graph is connected) but not everywhere differentiable. There is an ambiguity in the slope, with two choices at the articulation point between the two straight line segments.
One dimensional examples

The graph is not connected and so the field is not continuous and not differentiable.
Two dimensional examples

- The slope is dependent on the particular location and on the bearing at that location.
Isotropic fields

- A field whose properties are independent of direction is called an *isotropic field*

- Consider travel time in a spatial framework
  - The time from X to any point Y is dependent only upon the distance between X and Y and independent of the bearing of Y from X
Anisotropic fields

• A field whose properties are dependent on direction is called an *anisotropic field*.

• Suppose there is a high speed link $AB$
  
  – For points near $B$ it would be better, if traveling from $X$, to travel to $A$, take the link, and continue on from $B$ to the destination
  
  – *The direction to the destination is important*
Spatial autocorrelation

• Spatial autocorrelation is a quantitative expression of Tobler’s first law of geography (1970)
  – “Everything is related to everything else, but near things are more related than distant things”
  – Spatial autocorrelation measures the degree of clustering of values in a spatial field

• Also termed as spatial dependency, spatial pattern, spatial context, spatial similarity, spatial dissimilarity...
Autocorrelation

If there is no apparent relationship between attribute value and location then there is zero spatial autocorrelation.

If like values tend to be located away from each other, then there is negative spatial autocorrelation.

If like values tend to cluster together, then the field exhibits high positive spatial autocorrelation.
Representations of Spatial Fields

- Points
- Contours
- Raster/Lattice
- Triangulation (Delaunay Trangulation)
Example

- Contour lines and raster
Example

• Trangulations
Side Note: Delaunay Triangulation and Voronoi Diagram

- Dual Graph
Operations on fields

• A field operation takes as input one or more fields and returns a resultant field

• The system of possible operations on fields in a field-based model is referred to as *map algebra*

• Three main classes of operations
  – Local
  – Focal
  – Zonal
Neighborhood function

- Given a spatial framework $F$, a *neighborhood function* $n$ is a function that associates with each location $x$ a set of locations that are “near” to $x$.
Local operations

- **Local operation**: acts upon one or more spatial fields to produce a new field
- The value of the new field at any location is dependent on the values of the input field function at that location
  - is any binary operation
Focal operations

- **Focal operation**: the attribute value derived at a location \( x \) may depend on the attributes of the input spatial field functions at \( x \) and the attributes of these functions in the neighborhood \( n(x) \) of \( x \)
Zonal operations

- **Zonal operation**: aggregates values of a field over a set of zones (arising in general from another field function) in the spatial framework

- For each location $x$:
  1. Find the Zone $Z_i$ in which $x$ is contained
  2. Compute the values of the field function $f$ applied to each point in $Z_i$
  3. Derive a single value $\zeta(x)$ of the new field from the values computed in step 2
Summary: Object-based vs Field-based models

• Object-based models:
  – Greater precision
  – Less redundant information (smaller storage footprints)
  – Complex data structures

• Field-based models:
  – Simpler data structures
  – More redundant information (larger storage footprints)
  – Less precision

• *Raster is faster, but vector is corrector*
• End of this topic