Databases

Managing Data for Retrieval, Update, & Calculation

<table>
<thead>
<tr>
<th>Drilling Record</th>
<th>Production (barrels/day, cfs)</th>
<th>1997 Expenditures (millions of $s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
<td>Spudded</td>
</tr>
<tr>
<td>40</td>
<td>Exxon #1</td>
<td>2/4/96</td>
</tr>
<tr>
<td>43</td>
<td>Shell #5</td>
<td>3/14/97</td>
</tr>
<tr>
<td>72</td>
<td>1.501</td>
<td>0.652</td>
</tr>
<tr>
<td>72</td>
<td>5.522</td>
<td>0.301</td>
</tr>
</tbody>
</table>
GIS Data Recap

Two data types:

★ Spatial – *Where* things are, in \((x, y, \lambda, \phi)\)
  - Stored in coordinate files & topology *tables*
    - Vector (Object) Model
    - Raster (Field) Model

★ Aspatial - *What* things are
  - stored in *tables* of attributes
GIS = Lots and Lots of Tabular Data

How will it be managed?
- Data Model Considerations
- Analysis Considerations
- Data Entry Considerations
- Security
- Efficiency
HOW?

 Goals:

- Maximize flexibility for sorting, reordering, subsetting, searching
- Efficient storage; eliminate redundancy
- Secure entry and retrieval mechanisms
- Rapid retrieval

 Solution:

Database Management System (DBMS)
Database display in ArcMap

- Displayed in tables with rows of *records* (*tuples*) and columns of *fields* (*attributes*)
Accuracy

- Data entry can be accomplished via forms that require:
  - *Data definitions* – #s of attributes, the types and lengths or numerical ranges of each attribute, and how much editing will be permitted.
  - *Data Dictionary* - catalog of attributes with their permitted values and ranges (“Domains”).
  - *Validation Rules* - ensure data integrity.
Data Definition: Field “Types”

**Q:** how much space does a database need to reserve for each field?

**A:** no more than 10 characters
Fields are “defined” by:

- **Name** – attribute (column heading)
- **Field Type** – number (long, short, float, double), text ("string"), or date
- **Length** – no. of characters in text
- **Precision** – no. of digits used to store numbers
- **Scale** – no. of digits to right of decimal point

### Field Definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Precision</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>FID</td>
<td>Object ID</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shape</td>
<td>Point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDNUMBER</td>
<td>String</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PIT_TYPE</td>
<td>String</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PARISH</td>
<td>String</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CONTAINMT</td>
<td>String</td>
<td>51</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CONT_COND</td>
<td>String</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BREACHED</td>
<td>String</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RANKING</td>
<td>Long</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>
Fields Types in ArcGIS

- Short Integer – 1 to 4 digits (no decimal)
- Long Integer = 5 to 9 digits (no decimal)
- Float = 1 – 8 digits, decimal (short real)
- Double = 6 – 19 digits, decimal (long real)
- Text = 1 – 255 characters
- Date = 8 character
- Blob = binary large object
### Numeric Field Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Storage</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Integer</td>
<td>2 bytes</td>
<td>+/- 32,768</td>
<td>Used for coding, e.g. lulc, veg. types, T/F</td>
</tr>
<tr>
<td>Long Integer</td>
<td>4 bytes</td>
<td>+/- 2.14 billion</td>
<td>Large whole numbers, e.g. populations</td>
</tr>
<tr>
<td>Float</td>
<td>4 bytes</td>
<td>+/- 3.4 x E38</td>
<td>Single-precision, up to 6 places past the decimal. Up to 6 total numbers.</td>
</tr>
<tr>
<td>Double</td>
<td>8 bytes</td>
<td>~+/- 1.8 x E308</td>
<td>Double-precision; 15 places past decimal, 6-19 total numbers.</td>
</tr>
</tbody>
</table>
Field Properties

- Precision = number of digits stored in a field. Precision 7 for double, 1-6 for float
- Scale = no. of decimal places in double and float
- E.g. 3500426.21 should be stored as double, precision > 9, scale > 2
Numeric Field Properties

- Short and Long integers fields:
  Precision = 4
  8,400

- Float and Double data fields:
  Precision = 9
  8,400.08347
  Scale = 5
## File Size Comparison, Text Fields

<table>
<thead>
<tr>
<th>Text Field length</th>
<th>100 records</th>
<th>1000 records</th>
<th>10,000 records</th>
<th>100,000 records</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.2 Kb</td>
<td>1.95 Kb</td>
<td>19.53 Kb</td>
<td>195.3 Kb</td>
</tr>
<tr>
<td>50 (default)</td>
<td>4.88 Kb</td>
<td>48.83 Kb</td>
<td>488.3 Kb</td>
<td>4.88 Mb</td>
</tr>
</tbody>
</table>
Accuracy

✓ Data entry can be accomplished via *forms* that require:

✓ *Data definitions* – #s of attributes, the types and lengths or numerical ranges of each attribute, and how much editing will be permitted.

- *Data Dictionary* - catalog of attributes with their permitted values and ranges (“Domains”).

- *Validation Rules* - ensure data integrity.
Data Dictionary: Domains

- Permitted attribute values or range of values for a field:
  - E.g. dip of bedding: permissible range from 0-90°
  - E.g. type of geologic contact: permissibly covered, inferred, exposed
  - E.g. rock type: permissibly sandstone, shale, limestone
Domains in a Geodatabase

Geodatabase

Geodatabase Domains

Domain Values

Domain applied to Contacts05

Contacts05

Feature Class Fields
A DBMS provides:

✓ Accuracy - reduce errors during entry by use of established rules, templates

• Efficiency - rapid access & retrieval, no redundancy

• Flexibility - robust structure for query – e.g. What is where?

• Security – access and use can’t corrupt data

• Easy updating
Efficiency (+ Flexibility)

- Relies on database structure (data model):
  - Hierarchical
  - Network
  - Relational
  - Object-oriented
GIS attribute data models

- **Hierarchical** – pre-1980

- **Relational** – 1980’s, 1990’s; still dominant today

- **Object-oriented** – late ’90’s; newest, implemented by some GISs – still undergoing R&D
Hierarchical Structure

E.g. Filing cabinet or folders on a hard drive

- File address for storage and retrieval is a **linear path**, e.g.

  C:\ESRI\ESRIDATA\CANADA\cities.shp
Hierarchical - Limitations

1. Linear structure can’t deal with multiple “memberships”
   - E.g. a single well might be stored many times in different databases for taxes, production, drilling history, water quality, etc.

   INEFFICIENT

   - Can’t assemble all this data for query in a hierarchical database
Hierarchical - Limitations

2. Can’t deal with exceptions to linear scheme – entities may not belong to next higher class but could instead contain it.

   E.g. Structure Oil Well Database by:
   
   State
   County
   Oil Field
   Well
   Pay zone

What do we do with a oil field that spans several counties with wells that produces from more than one pay zone?

   ✿ i.e. No “one-to-many” relationships
Relational Database Advantages

- Data stored in separate tables
  - Easy update, editing, searching without affecting or using all data

- Flexibility
  - Using “key” fields, can extract and assemble records and attributes to form new tables
  - Subsets of database can be queried by standard means - SQL
Relational Database Structure

- Consists of “relations” (tables) with multiple attributes (columns, fields) per record
- Every record (row) has a unique identifier (marker or key attribute)
  - Key is the glue between files that can be used to extract and/or assemble records and attributes
## Parts of a Relation

### Primary Key
- ID

### Record or Tuple
- ID = 40: Oil = 53, Gas = 1200, Water = 5
- ID = 55: Oil = 108, Gas = 2500, Water = 15

### Attribute or Field
- Production (barrels/day, cfs)

<table>
<thead>
<tr>
<th>ID</th>
<th>Oil (barrels/day)</th>
<th>Gas (cfs)</th>
<th>Water (cfs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>53</td>
<td>1200</td>
<td>5</td>
</tr>
<tr>
<td>55</td>
<td>108</td>
<td>2500</td>
<td>15</td>
</tr>
</tbody>
</table>
Properties of Relations

- Each row has to be unique; no row-to-row dependency
- Row order irrelevant
- Column order irrelevant
- All attribute values must be stored in separate rows ("first normal form")
Relational Database Structure

<table>
<thead>
<tr>
<th>Drilling Record</th>
<th>ID</th>
<th>Name</th>
<th>Spudded</th>
<th>Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Exxon #1</td>
<td>2/4/96</td>
<td>6/3/96</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Shell #5</td>
<td>3/14/97</td>
<td>6/12/96</td>
<td></td>
</tr>
</tbody>
</table>

“Key” Field = ID

1997 Expenditures (millions of $s)

<table>
<thead>
<tr>
<th>ID</th>
<th>Drilling</th>
<th>Production</th>
<th>Transport.</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.501</td>
<td>0.652</td>
<td>0.078</td>
</tr>
<tr>
<td>72</td>
<td>5.522</td>
<td>0.301</td>
<td>0.055</td>
</tr>
</tbody>
</table>

Production (barrels/day, cfs)

<table>
<thead>
<tr>
<th>ID</th>
<th>Oil</th>
<th>Gas</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>53</td>
<td>1200</td>
<td>5</td>
</tr>
<tr>
<td>43</td>
<td>108</td>
<td>2500</td>
<td>15</td>
</tr>
</tbody>
</table>
One record from *source table* (production) is joined to one record of *destination table* (drilling record) to create a “View” – virtual combination
Result of One-to-One Table “Join”

Joined Production and Drilling Record tables:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Spudded</th>
<th>Completed</th>
<th>Oil</th>
<th>Gas</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Exxon #1</td>
<td>2/4/96</td>
<td>6/3/96</td>
<td>53</td>
<td>1200</td>
<td>5</td>
</tr>
<tr>
<td>43</td>
<td>Shell #5</td>
<td>3/14/97</td>
<td>6/12/96</td>
<td>108</td>
<td>2500</td>
<td>15</td>
</tr>
</tbody>
</table>

View can’t be edited – destination table can be
### One-to-Many Join

**Drilling Record**

<table>
<thead>
<tr>
<th>ID</th>
<th>Well Name</th>
<th>Spudded</th>
<th>Completed</th>
<th>Field_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Exxon #1</td>
<td>2/4/96</td>
<td>6/3/96</td>
<td>2</td>
</tr>
<tr>
<td>43</td>
<td>Shell #5</td>
<td>3/14/95</td>
<td>6/12/96</td>
<td>2</td>
</tr>
<tr>
<td>72</td>
<td>Amoco #3</td>
<td>4/8/88</td>
<td>4/8/89</td>
<td>2</td>
</tr>
<tr>
<td>55</td>
<td>BP #2</td>
<td>6/8/90</td>
<td>8/8/91</td>
<td>Wildcat</td>
</tr>
</tbody>
</table>

**Oil/Gas Fields**

<table>
<thead>
<tr>
<th>Field_ID</th>
<th>Name</th>
<th>Discovered</th>
<th>Total_Oil</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Longview</td>
<td>1/20/56</td>
<td>13000564</td>
</tr>
<tr>
<td>2</td>
<td>Katy</td>
<td>2/3/48</td>
<td>85640</td>
</tr>
<tr>
<td>3</td>
<td>Anhuac</td>
<td>4/11/73</td>
<td>3587889</td>
</tr>
</tbody>
</table>

※One record from source table joined to many records of destination table.
### One-to-Many Join Result

#### Table of Wells in Katy Field

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Spudded</th>
<th>Completed</th>
<th>Field_ID</th>
<th>Name</th>
<th>Discovered</th>
<th>Total_Oil</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Exxon #1</td>
<td>2/4/96</td>
<td>6/3/96</td>
<td>2</td>
<td>Katy</td>
<td>2/3/48</td>
<td>85640</td>
</tr>
<tr>
<td>43</td>
<td>Shell #5</td>
<td>3/14/95</td>
<td>6/12/96</td>
<td>2</td>
<td>Katy</td>
<td>2/3/48</td>
<td>85640</td>
</tr>
<tr>
<td>55</td>
<td>BP #2</td>
<td>6/8/90</td>
<td>8/8/91</td>
<td>Wildcat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note that “Discovered” and “Total_Oil” fields in joined table pertain to Katy Field, not to individual wells! This is a problem...*
Many-to-Many Join example – USGS DLGs

Join “lookup table” with feature codes tables to obtain feature descriptions

- Feature descriptions stored once, used many times
- Primary key is feature code
Digital Line Graph Example

Lookup Table

Key

Hydrography feature attributes
Result of Many-to-Many Join

- Symbolize on joined field (FEAT_DESC)

Layer = Condrey_HY
A DBMS provides:

- **Accuracy** - reduce errors during entry by use of established rules, templates
- **Efficiency** - rapid access & retrieval, no redundancy
- **Flexibility** - robust structure for query – e.g. What is where?
- **Security** – access and use can’t corrupt data
- **Easy updating**
Flexibility

zeń Using primary key(s), can extract and assemble records and attributes to form new tables, as discussed

ז Subsets of database can be queried by standard means - SQL
Example: Find all cities in Louisiana where 1990 population exceeded 72,033.
Relational DBMSs Permits:

- File updating
- Data retrieval via query using a standard language (SQL)
- Sorting (reordering) by field values
- Calculations and field statistics
- Report generation
- Multi-user access
Reordering In ArcMap

1. Order selected records by sorting
   - ascending or descending field values
2. Sort records by selected attributes

Unsorted

Sorted by “Azimuth”
Field Statistics In ArcMap

- Get stats. & graphs on selected attributes

Statistics for “Azimuth”

Histogram for “Dip”
GIS’ are Spatial Databases

Coverage and Shapefile models

- Spatial information stored in spatial attribute files, attributes in relational database table
  - Feature ID is key
  - Spatial information can’t participate in relational database advantages

Geodatabase model

- All information, spatial and aspatial, are stored together in a relational database