

Databases



SU 5050
LECTURE 6
JESSICA L. MCCARTY, PH.D.

Process



- | | |
|---|-------------|
| 1. Develop understanding of application, goals | Databases |
| 2. Create dataset | |
| 3. Data cleaning and preprocessing | |
| 4. Data reduction and projection | Data Mining |
| 5. Choose data mining task | |
| 6. Choose data mining algorithms | |
| 7. Use algorithms to perform task | |
| 8. Interpret and iterate thru 1-7 <i>if necessary</i> | |
| 9. Deploy: integrate into/create new operational system | |

Data Mining Tasks



- Classification *[Predictive]*
- Clustering *[Predictive]*
- Association Rule Discovery *[Descriptive]*
- Sequential Pattern Discovery *[Descriptive]*
- Natural Language Processing *[Descriptive]*
- Regression *[Predictive]*
- Deviation Detection *[Predictive]*

What is Data?

- Collection of data objects and their attributes
- An attribute is a property or characteristic of an object
 - Examples: eye color of a person, temperature, etc.
 - Attribute is also known as variable, field, characteristic, or feature
- A collection of attributes describe an object
 - Object is also known as record, point, case, sample, entity, or instance

Attributes

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Objects

Types of Attributes



- There are different types of attributes
 - **Nominal**
 - ◆ Examples: ID numbers, eye color, zip codes
 - **Ordinal**
 - ◆ Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
 - **Interval**
 - ◆ Examples: calendar dates, temperatures in Celsius or Fahrenheit.
 - **Ratio**
 - ◆ Examples: temperature in Kelvin, length, time, counts

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. (=, ≠)	zip codes, employee ID numbers, eye color, sex: { <i>male</i> , <i>female</i> }	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects. (<, >)	hardness of minerals, { <i>good</i> , <i>better</i> , <i>best</i> }, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, <i>t</i> and <i>F</i> tests
Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Attribute Level	Transformation	Comments
Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
Ordinal	An order preserving change of values, i.e., $new_value = f(old_value)$ where f is a monotonic function.	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Interval	$new_value = a * old_value + b$ where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
Ratio	$new_value = a * old_value$	Length can be measured in meters or feet.

Discrete and Continuous Attributes



- **Discrete Attribute**

- Has only a finite or countably infinite set of values
- Examples: zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables.
- Note: binary attributes are a special case of discrete attributes

- **Continuous Attribute**

- Has real numbers as attribute values
- Examples: temperature, height, or weight.
- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.

Types of Data Sets



- **Record**

- Data Matrix
- Document Data
- Transaction Data

- **Graph**

- World Wide Web
- Molecular Structures

- **Ordered**

- Spatial Data
- Temporal Data
- Sequential Data
- Genetic Sequence Data

Datasets: Text Files



- <http://www.iana.org/assignments/character-sets/character-sets.xhtml>
 - Under Class URLs on Canvas
- Character sets for the internet
- 600+ character sets are available for download as:
 1. .xml
 2. .html
 3. .txt
 4. .CSV

ASCII Text Files



- ASCII (American Standard Code for Information Exchange)
 - English alphabet
 - 128 possible characters
 - <http://www.network-science.de/ascii/>
 - ASCII Art

ASCII TABLE

Hexadecimal, base 16, to is a positional numeral system.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

XML



- Extensible Markup Language
 - Uses tags to describe data elements
- Defined by W3c (World Wide Web Consortium)
 - For data exchange over networks
- Few predefined elements
 - Minimalist
- Tree structure
 - Parent nodes, subnodes

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="chapter">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sentence" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="title" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="report">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="toc"/>
        <xs:element ref="chapter" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="sentence">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="ref" type="xs:string"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="toc">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tocitem" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="tocitem">
    <xs:complexType>
      <xs:attribute name="chapter" type="xs:IDREF" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML



- Elements in XML described by a DTD
 - Document Type Definition
- Describe elements
 - Name
 - Type
 - Format
 - Order

XML



- XML is **well-formed** if
 - Element tags are matched
 - The tags are closed correctly
- XML document is **valid** if
 - Structure conforms to the DTD

HTML



- HyperText Markup Language
 - Text/html
 - More during web scraping

```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
  <title>sample</title>
</head>
<body>
  <p>Voluptatem accusantium
  totam rem aperiam.</p>
</body>
</html>
```

HTML

.TXT

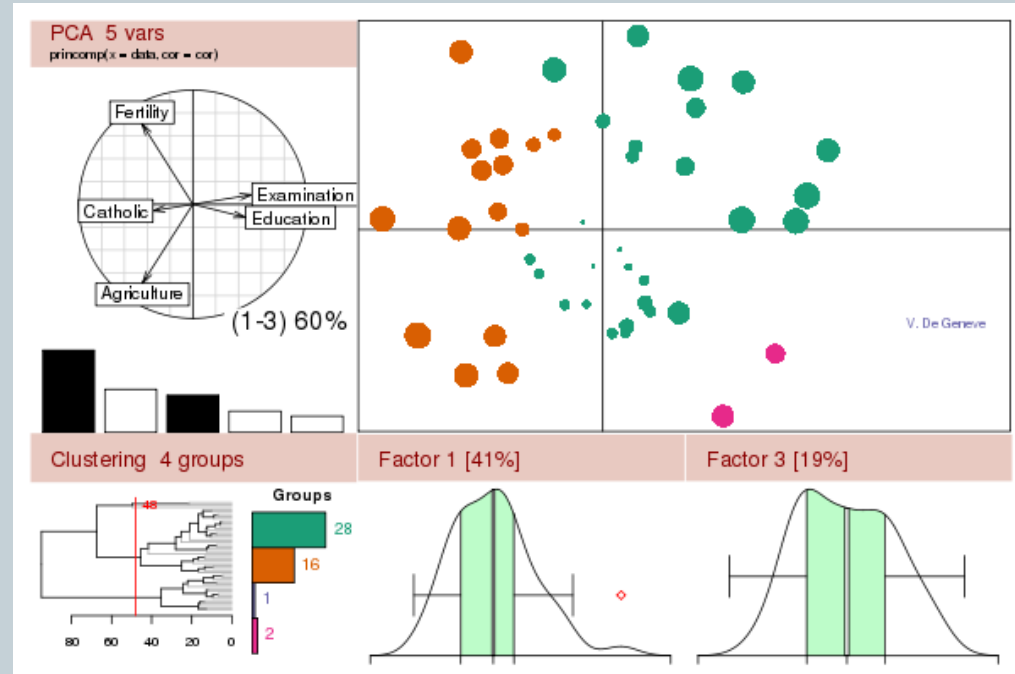


- Text files with little formatting (no bolding or italics)
- Format not defined
 - Determined by system (Windows, LINUX, Mac)
 - Text Editor (e.g. TextPad, Notepad, WordPad)
- Easily read by any program that reads text
 - Considered universal/platform independent

.DAT



- Not a specified file type
- Generic extension for “data”
- Input, Output for R
 - <http://www.r-project.org/>
 - http://anson.ucdavis.edu/~abshev/teaching/12winter/Getting_Data_in_to_R.pdf



.CSV



- Comma Separated Values – most common import, export for spreadsheets and databases
- No standard “CSV”
 - Software dependent
 - Platform dependent
 - Python has CSV module and API
 - <https://docs.python.org/2/library/csv.html>

.DBF4



- Excel, Access, SQL database type that is read by all commercial GIS (i.e., ESRI) and opensource (i.e., qgis, gdal)
- When manipulating data for later geospatial applications within GIS, be able to transfer between dbf4 (dbf IV)

Python File Formats



- Stick with CSV
- Or creating your own lists (time consuming)
- <https://docs.python.org/2/library/fileformats.html>

What is a Database?



- “A set of information held in a computer”

Oxford English Dictionary

- “One or more large structured sets of persistent data, usually associated with software to update and query the data”

Free On-Line Dictionary of Computing

- “A collection of data arranged for ease and speed of search and retrieval”

Dictionary.com

Databases



- Web indexes
- Library catalogues
- Medical records
- Bank accounts
- Stock control
- Personnel systems
- Product catalogues
- Telephone directories
- Train timetables
- Airline bookings
- Credit card details
- Student records
- Customer histories
- Stock market prices
- Discussion boards
- and so on...

Database Systems



- A database system consists of

- Data (the database)
- Software
- Hardware
- Users

- Database systems allow users to

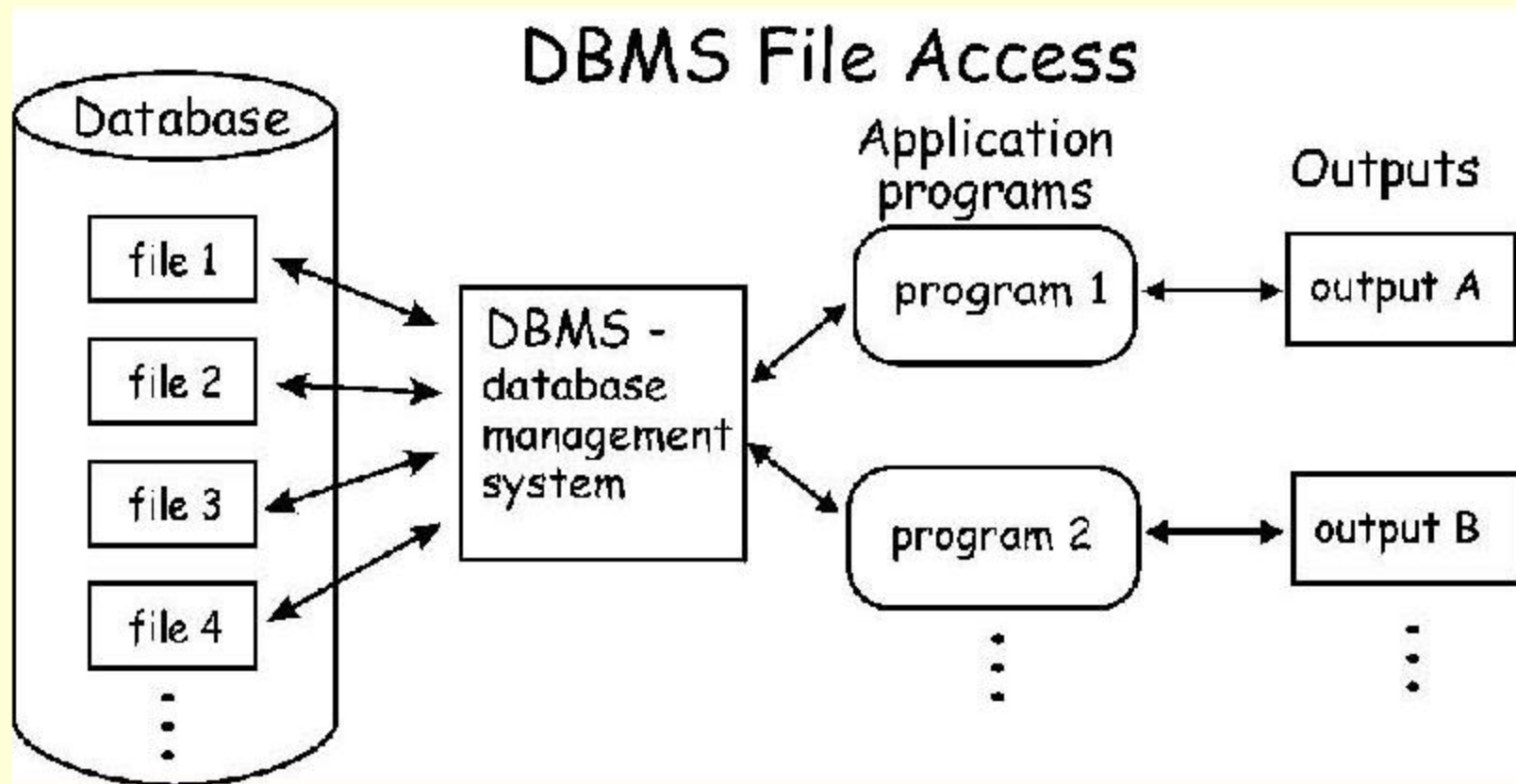
- Store
- Update
- Retrieve
- Organise
- and Protect their data.

Database Management Systems



- A database is a collection of information
 - A database management system (DBMS) is the software that controls that information
- Examples:
 - Oracle
 - DB2 (IBM)
 - MS SQL Server
 - MS Access
 - Ingres
 - PostgreSQL
 - MySQL

DBMS controls access



What the DBMS does



- Provides users with
 - Data definition language (DDL)
 - Data manipulation language (DML)
 - Data control language (DCL)
- Often these are all the same language.
 - In ArcGIS, the DDL is called the Data Definition Language

- DBMS provides
 - Persistence
 - Concurrency
 - Integrity
 - Security
 - Data independence
- Data Dictionary
 - Describes the database itself

Data Dictionary - Metadata

- The dictionary or catalog stores information about the database itself
 - This is data about data or 'metadata'
 - Almost every aspect of the DBMS uses the dictionary
- The dictionary holds
 - Descriptions of database objects (tables, users, rules, views, indexes,...)
 - Information about who is using which data (locks)
 - Schemas and mappings



File Based Systems



- **File based systems**

- Data is stored in files
- Each file has a specific format
- Programs that use these files depend on knowledge about that format

- **Problems:**

- No standards
- Data duplication
- Data dependence
- No way to generate ad hoc queries
- No provision for security, recovery, concurrency, etc.

Relational Systems



- Problems with early databases
 - Navigating the records requires complex programs
 - There is minimal data independence
 - No theoretical foundations
- Then, in 1970, E. F. Codd wrote “A Relational Model of Data for Large Shared Databanks” and introduced the relational model.

Relational Systems



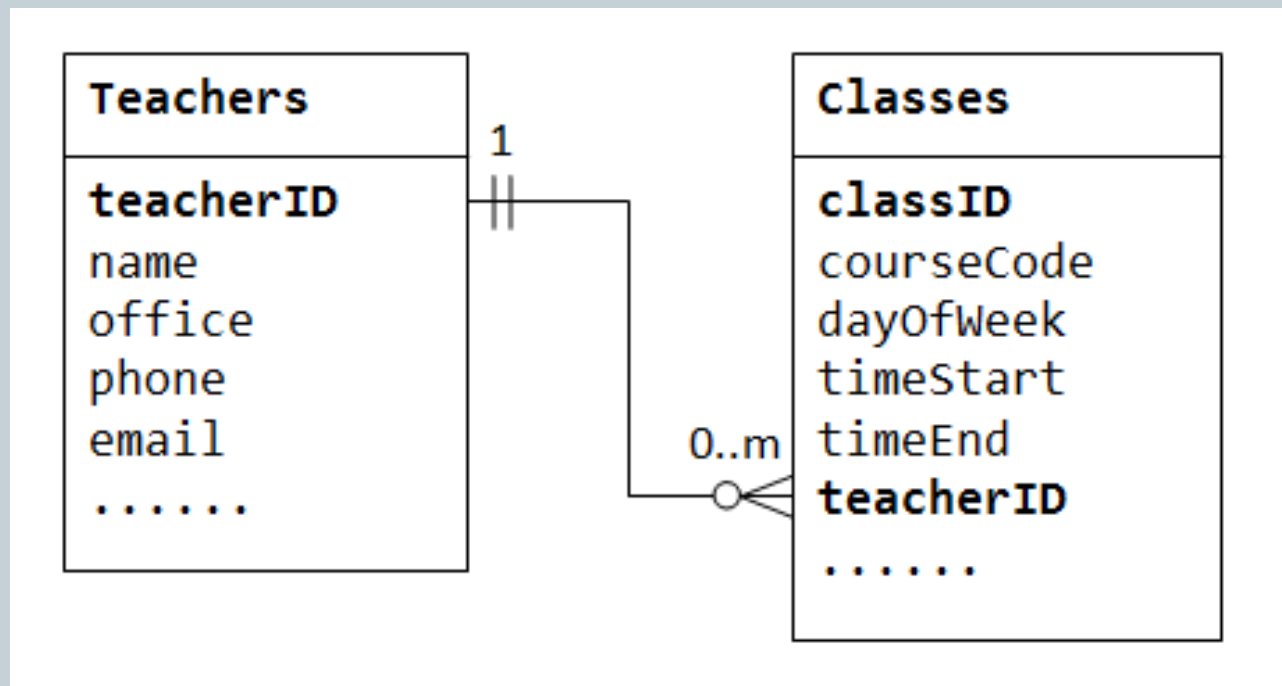
- Information is stored as *tuples* or *records* in *relations* or *tables*
 - There is a sound mathematical theory of relations
 - Most modern DBMS are based on the relational model
- The relational model covers 3 areas:
 - Data structure
 - Data integrity
 - Data manipulation

Relationships among Data Tables

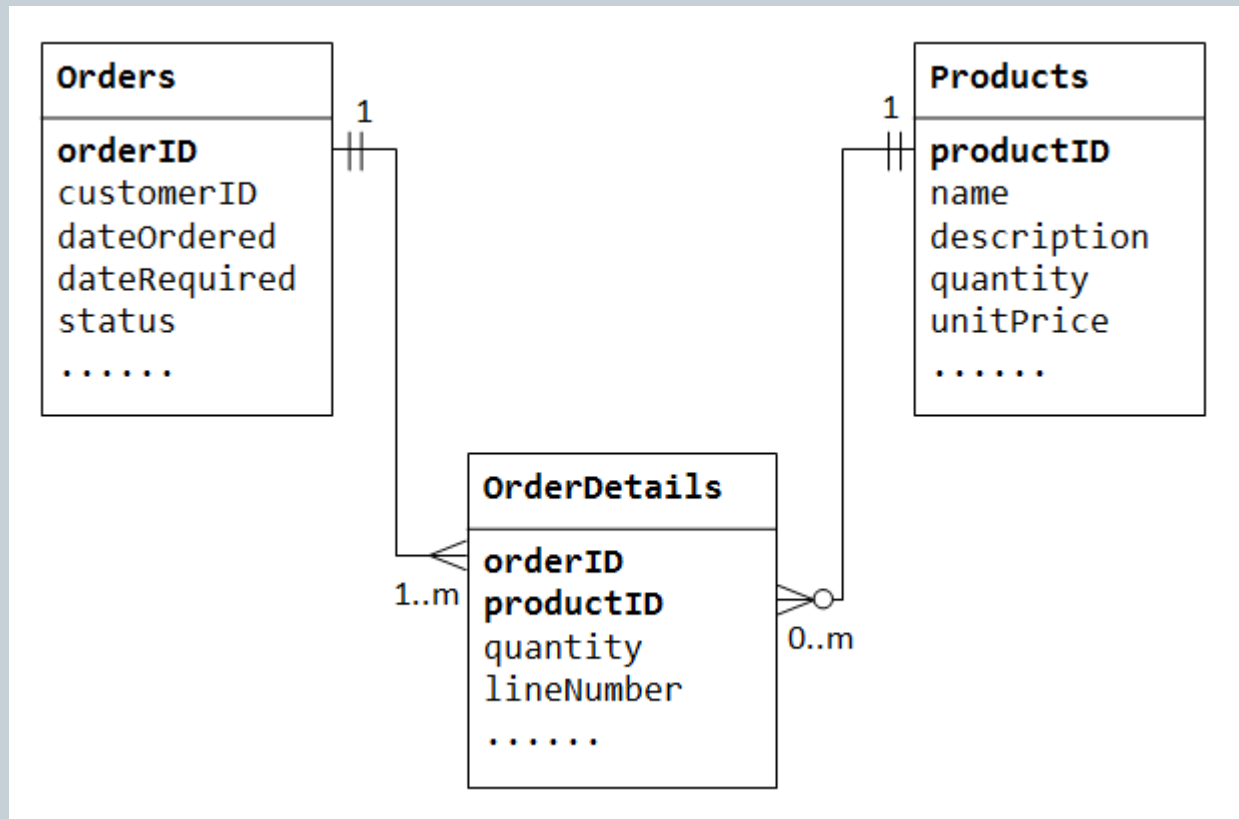


1. One-to-many
2. Many-to-many
3. One-to-one

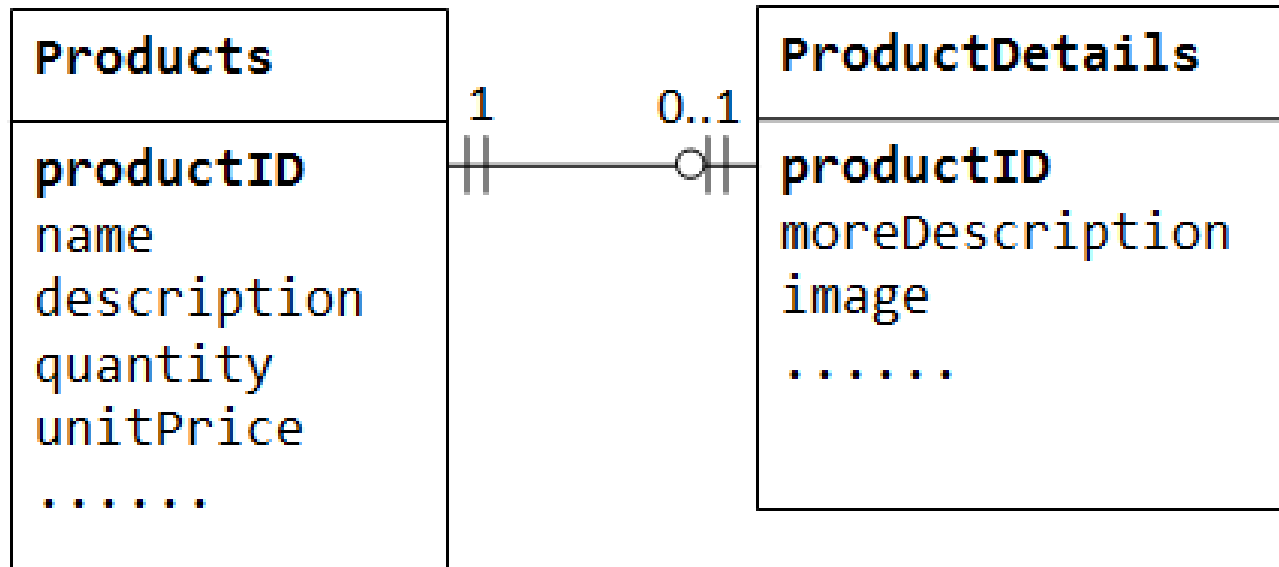
One-To-Many



Many-to-Many



One-to-One



ANSI/SPARC Architecture



- ANSI - American National Standards Institute
 - SPARC - Standards Planning and Requirements Committee
 - 1975 - proposed a framework for DBs
- A three-level architecture
 - Internal level: For systems designers
 - Conceptual level: For database designers and administrators
 - External level: For database users

Internal Level



- Deals with physical storage of data
 - Structure of records on disk - files, pages, blocks
 - Indexes and ordering of records
 - Used by database system programmers

- Internal Schema

```
RECORD EMP
LENGTH=44
HEADER: BYTE (5)
        OFFSET=0
NAME:   BYTE (25)
        OFFSET=5
SALARY: FULLWORD
        OFFSET=30
DEPT:   BYTE (10)
        OFFSET=34
```

Conceptual Level



- Deals with the organization of the data as a whole
 - Abstractions are used to remove unnecessary details of the internal level
 - Used by DBAs and application programmers

- Conceptual Schema

```
CREATE TABLE  
Employee (  
    Name  
        VARCHAR(25) ,  
    Salary REAL,  
    Dept_Name  
        VARCHAR(10) )
```

External Level



- Provides a view of the database tailored to a user
 - Parts of the data may be hidden
 - Data is presented in a useful form
 - Used by end users and application programmers

- External Schemas

Payroll:

String Name

double Salary

Personnel:

char *Name

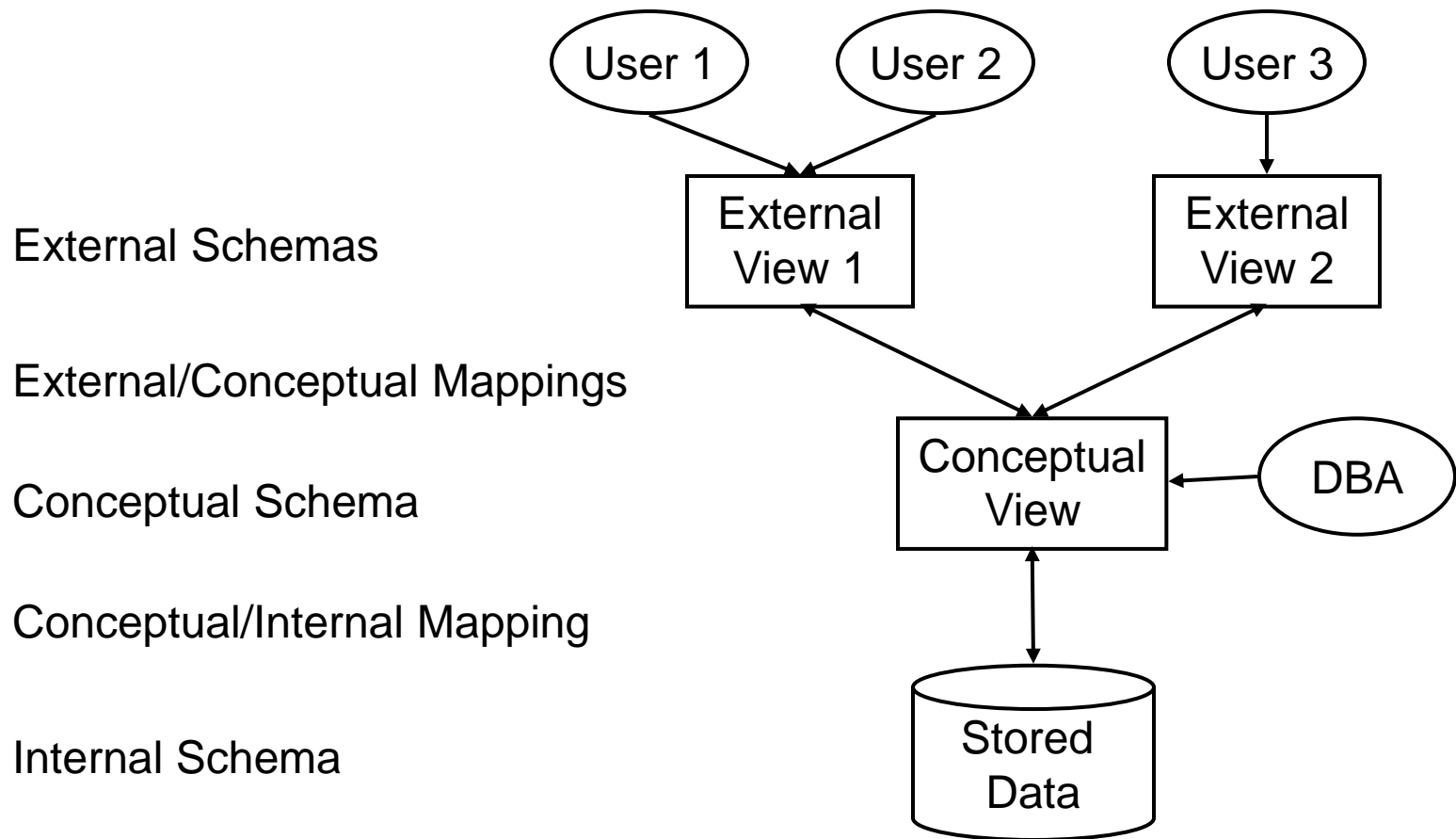
char *Department

Database Mappings



- Mappings translate information from one level to the next
 - External/Conceptual
 - Conceptual/Internal
 - These mappings provide data independence
- Physical data independence
 - Changes to internal level shouldn't affect conceptual level
 - Logical data independence
 - Conceptual level changes shouldn't affect external levels

ANSI/SPARC Architecture





Example of Geospatial DBMS

A DBMS Contains






1. Data definition language
 - a) DDL in ArcGIS
2. Data dictionary
3. Data entry module
4. Data update module
5. Report generator
 - a) Reporting tools – a report, menu, wizard, viewer, designer (.rlf or .rdf files)
6. Query language
 - a) SQL in ArcGIS [Structured Query Language]

What Is A Data Dictionary?

DATA DICTIONARY

Learner Table

P/F	Field Name	Caption	Data Type	Field Size	Notes
P	learner_id		Autonumber		
F	instructor_id	Usual Instructor	Long Integer		
	str_first_name	First Name	Text	20	
	str_last_name	Last Name	Text	30	
	str_house_no	House Number/Name	Text	20	
	str_street	Street	Text	30	
	str_locality	Locality	Text	30	
	str_town	Town	Text	30	
	str_county	County	Text	30	
	str_post_code	Post Code	Text	9	>LL09 90LL

			
ArcGIS Product	Included with ArcEditor and ArcInfo Included with ArcGIS Engine	Included with ArcGIS Server Workgroup	Included with ArcGIS Server Enterprise
Number of ArcSDE geodatabase users	Max. 3 Users	Max. 10 Users at a time No limit on the number of connections from servers	Unlimited
Supported DBMSs	SQL Server Express 1GB RAM 1 CPU	SQL Server Express 1GB RAM 1 CPU	SQL Server Oracle IBM DB2 Informix PostgreSQL
Database size limits	Max database size 4GB	Max database size 4GB	No limits

Use of DBMS with GIS

- *standalone GIS desktop* – largely focused on doing “one-off” GIS projects (**personal geodatabase**)
- *desktop GISs using shared DBMS database servers*
- *server-centric GISs* – solutions where GIS applications are supported centrally with (thin-)client access (see arcgis.com)
- *GIS networks* – federations of coupled servers that support Web services standards and integrated with portals
- *Mobile GISs* on a variety of personal devices (often integrated with GPS) – they can be wireless or standalone systems and are often part of one of the above.

Advantages of DBMS

- *Simultaneous multi-user access* – DBMS manages multiple versions.
- *Data independence* – data are structured independently of the applications that depend on them.
- *Multiple user views* – different applications can see the data in different formats.
- *Centralized control and maintenance* – assures integrity and efficiency.

Relational DBMS

- Separate types of entities are stored in separate tables
- Each table has a “key” variable that uniquely identifies each entity (can be system or user defined)

Forests

Forest Name	Location	Size
Nantahala	North Carolina	184,447
Cherokee	North Carolina	92,271

Recreation Features

Feature	Description	Activities
Wfall	Waterfall	Photography, Swimming
Ogrth	Old-Growth Forest	Photography, Hiking
Vista	Scenic overlook	Photography, viewing
Wlife	Wildlife Viewing	Photography, Birding
Cmp	Camping	Camping

Trails

Trail Name	Difficulty	Forest	Feature
Bryson's Knob	E, M	Nantahala	Vista, Ogrth
Slickrock Falls	M	Cherokee	Wfall, Ogrth
North Fork	M	Nantahala	-
Cade's Cove	E	Cherokee, Nantahala	Ogrth, Wlife
Appalachian	M, D	Nantahala, Cherokee	Wfall, Ogrth, Vista, Wlife, Cmp

Relational Joins

Example

- Relational Joins can be used to link any two tables that share a common key. **Unique ID**
- User or database creator sets up joins.
- Then, tables can be treated as one, though store separately.

POLYGON

Site #	Area	VegType	SoilType	Elevation
1	1342	A	1	650
2	14567	C	3	760
3	18378	B	4	634
4	1312	A	3	590

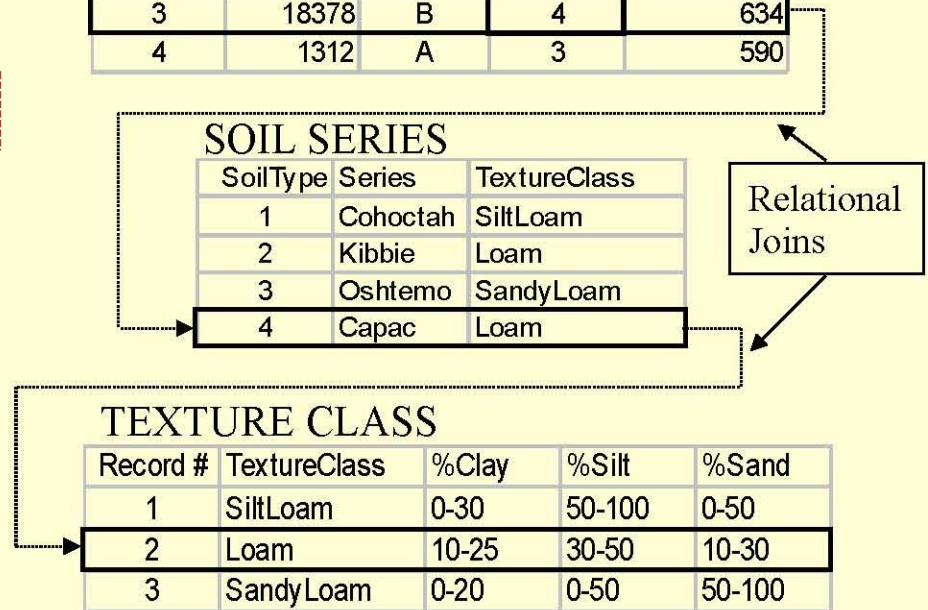
SOIL SERIES

SoilType	Series	TextureClass
1	Cohoctah	SiltLoam
2	Kibbie	Loam
3	Oshtemo	SandyLoam
4	Capac	Loam

TEXTURE CLASS

Record #	TextureClass	%Clay	%Silt	%Sand
1	SiltLoam	0-30	50-100	0-50
2	Loam	10-25	30-50	10-30
3	SandyLoam	0-20	0-50	50-100

Relational
Joins



Normalization

- It is more efficient to store information in separate files.
- *Normalization* is the process of identifying redundancies in the tables and breaking them into separate tables.

- Table with redundancy

Forest Name	Forest-ID	Location	Size	Trail Name
Nantahala	1	N. Carolina	184,447	Bryson's Knob
Nantahala	1	N. Carolina	184,447	North Fork
Nantahala	1	N. Carolina	184,447	Cade's Cove
Nantahala	1	N. Carolina	184,447	Appalachian
Cherokee	2	N. Carolina	92,271	Slickrock Falls
Cherokee	2	N. Carolina	92,271	Cade's Cove
Cherokee	2	N. Carolina	92,271	Appalachian

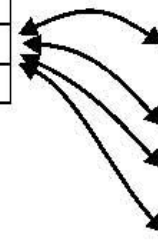
- Normalized tables with separate tables for each separate entity.

Forests

Forest Name	Forest-ID	Location	Size
Nantahala	<u>1</u>	N. Carolina	184,447
Cherokee	2	N. Carolina	92,271

Trails

Trail Name	Forest-ID
Bryson's Knob	<u>1</u>
Slickrock Falls	2
North Fork	<u>1</u>
Cade's Cove	<u>1</u>
Cade's Cove	2
Appalachian	1
Appalachian	2



Object-Relational Model

- Combines relational tables with more complex features of “object-orientation.”
- Allows fields to have more elaborate information and descriptions.
- Special fields include:
 - “Geometry” field of a table can store spatial features (i.e., objects).
 - BLOB (binary large object) field type is also available
 - Can store pictures, video, sound.

Python and ArcMap



**MORE GEOSPATIAL DATA MINING
CONSIDERATIONS**

Benefits of Python



- Easy to read and use syntax
- Large library
- Supports raising and catching exceptions
- Supports objects oriented programming

Python and ArcMap



- Python is used to automate geoprocessing tools such as the following which are in the ArcToolbox:
 - Analysis, cartography, conversion, data management, editing, geocoding
- Python also allows more advanced processing such as **looping through records in a database** and reading and writing them
- It also allows **manipulating layers in a map**
- It allows **creating and manipulating geometries** (point, line, polygon)

ArcPy



- arcpy was introduced to ArcGIS in its 10th version
 - It is downloaded with your ArcGIS!
 - You can find it under your ArcGIS folder, e.g. at:
Program Files > ArcGIS > Desktop10.2.2
- You just need an editor such as PythonWin, IDLE, ArcMap Python window, or even Calculator field to access arcpy
- If after typing import arcpy in front of the Python prompt you do not get an error, then you have the arcpy!

```
>>> import arcpy
```

```
# Got no error? Then you can slither!
```

ArcPy Opens Modules



- **arcpy can open many modules**
 - **Module:** is a self contained collection of functions and classes that does something
- **These modules include:**
 - Data access module (**arcpy.da**)
 - Mapping module (**arcpy.mapping**)
 - Geostatistical Analyst module (**arcpy.ga**)
 - ArcGIS Spatial Analyst extension module (**arcpy.sa**)
 - ArcGIS Network Analyst extension module (**arcpy.na**)

ArcPy



```
import arcpy # imports ArcGIS geoprocessing functionality
```

```
import arcpy.mapping # imports only the mapping module
```

```
import os # imports Python's core operating system
```

```
import sys # variables/functions used or maintained by the interpreter
```

```
# import env from arcpy and set the workspace environment
```

```
from arcpy import env # ability to control ArcGIS  
environment
```

```
env.workspace = "C:\data"
```

```
from arcpy.management import *
```

```
# Content imported into namespace. Can use content without  
prefix
```

Python Window



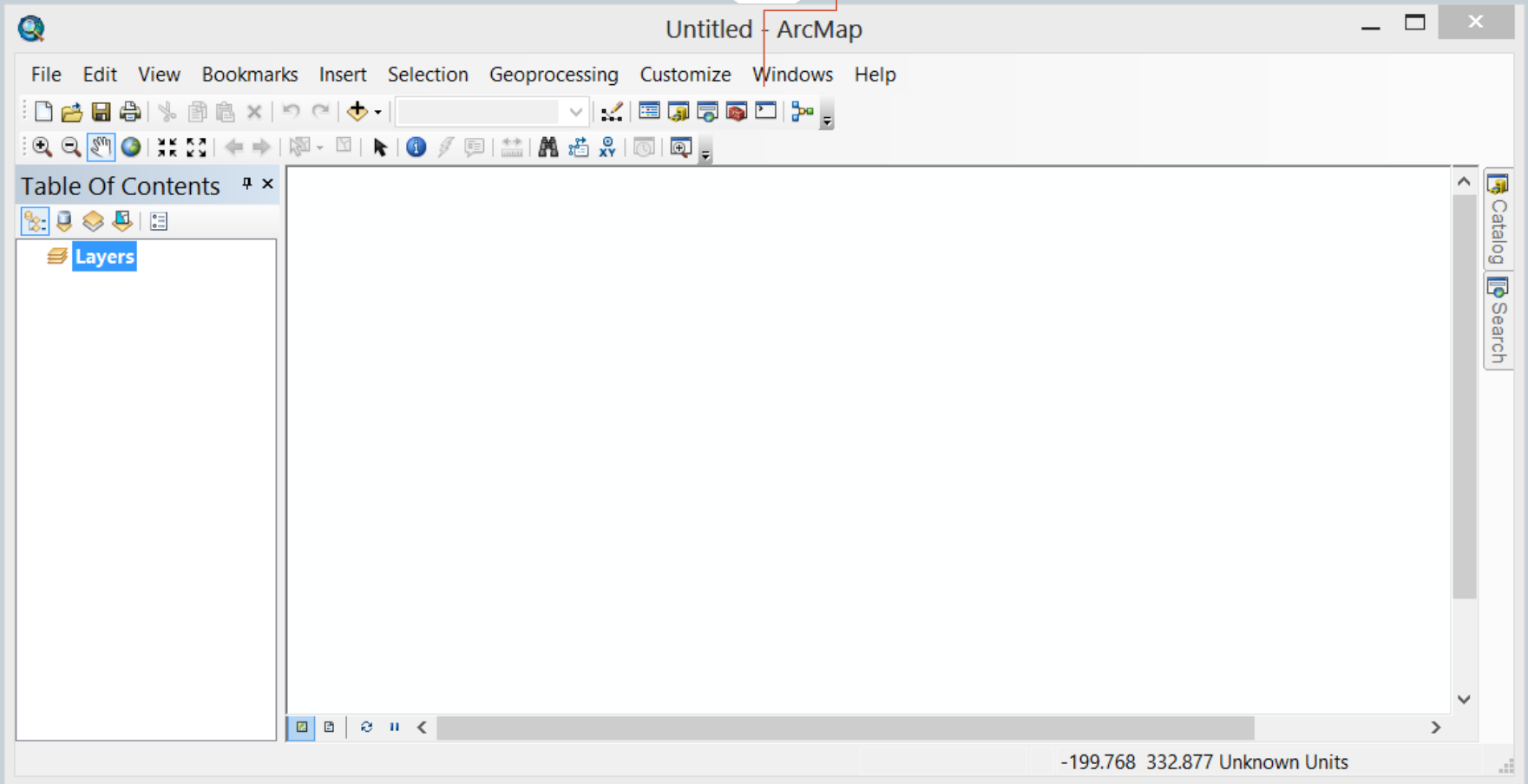
- Interacts with ArcGIS
- Accesses arcpy and its functionalities
- Accesses tools, and environments
- It is intelligent, completes code

Running in ArcMap

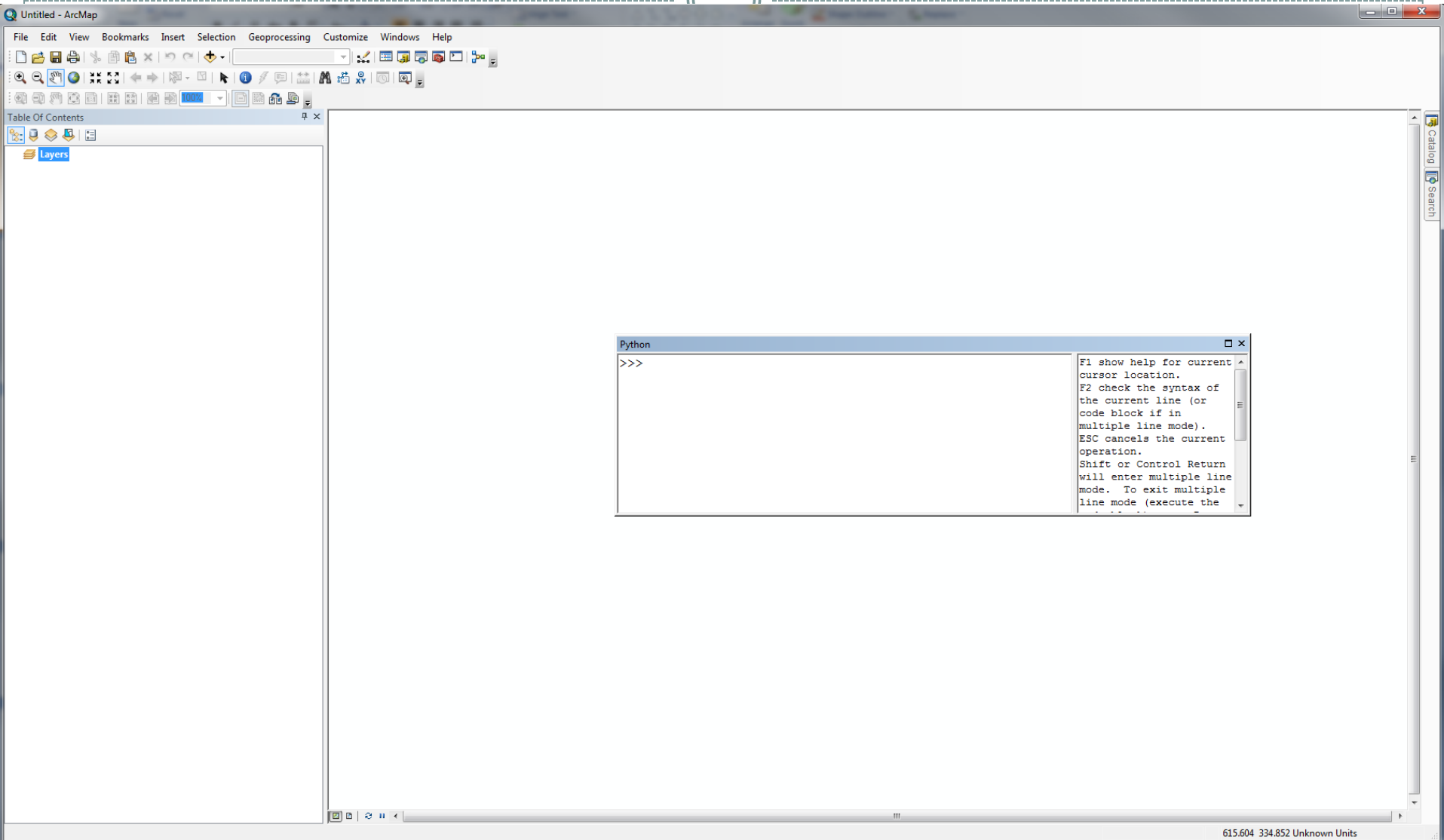


- Open **Python Window** in version 10.# by clicking on its icon in the menu bar
- We can write Python code while a map is open, for example to set a buffer around a road at different distances
- It will automatically create new layer for these buffers (e.g., at different distances)

Python Window



Python in ArcMap



ArcObjects



- ArcGIS is made of many types of **ArcObjects**
- These include: features, layers, maps, map documents, applications
- Even tables, their fields, and rows are ArcObjects
- Each of these ArcObjects has its own properties and methods, through which it interacts with other ArcObjects
- **ArcObjects can be manipulated with arcpy**

Properties and Methods



- Manipulating ArcObjects requires knowing their properties
- Properties/Methods of two ArcObjects are shown below:

Map

Properties

layer count
Name
Spatial reference
Map scale
Extent

Methods

Add layer
Clear Selection
Select feature

Feature Class

Properties

Shape type
Spatial reference
Extent

Methods

Create feature
Remove feature

Statements



- **A statement is a task that does not return a value, e.g., print, import, and if statements:**

```
import arcpy
```

```
print "bye; have a gneiss rock!"
```

```
# import functionality from ArcGIS Spatial Analyst
```

```
import arcpy.sa
```

```
from arcpy.sa import *
```

```
arcpy.CheckOutExtension ("Spatial")    # Check the Spatial Analyst License
```

```
import arcpy.mapping as map
```

- If statement checks if a condition is true or false
- For loop is another statement, loops through a list

```
for fc in fclist:
```

```
    print fc
```

Environments



- We set the environment for tools to use them
- This includes setting the current workspace, output spatial reference, extent, raster analysis setting (cell size, mask)

`arcpy.env.workspace`

`arcpy.env.outputCoordinateSystem`

`arcpy.env.extent`

`arcpy.env.cellSize`

`arcpy.env.mask`

Example 1: buffer



```
# add buffer around the road feature class with given distances
import arcpy
arcpy.env.workspace = "C:\data\City.gdb"           #sets the workspace
fc = "Roads"                                       #variable feature class
distanceList = ["100 meters", "200 meters", "400 meters"] # distances

# loops through each distance in the distanceList
# takes the first distance and puts it in variable dist, and repeats it 3 times
for dist in distanceList:
    outName = fc+"_" + dist
    arcpy.Buffer_analysis (fc, outName, dist)
# outputs the feature class, its output name_distance
# breaks out of the loop
print "Buffering completed!"
```

Example 2: clipping



- Clipping a road feature in ArcMap
- Click on the Python Window icon in ArcMap
- Clear the window from old code, if necessary
- Type the following code:

```
# clip Roads feature class from the Mineral_Spring city polygon,  
# output the clipped feature class into RoadsClip  
arcpy.clip_analysis ("Roads", "Mineral_Springs", "RoadsClip")  
  
# Run it  
# You can see in ArcMap the new added RoadsClip feature class  
# which was created and automatically added to the table of content
```


Adding a Python script as a tool



- We can add a script as a tool to a toolbox
- These are called **script tools**
- They become a new tool with all the properties of a tool, e.g.,
 - It will return messages, access to all environment settings, and automatically add the output to our map (to the table of contents in ArcMap)
- Can easily be shared, e.g., email it to other users
 - they can double click it to run it)
- Tools automatically create dialog boxes (created for us by ArcGIS)
- We can even add the tool into the toolbar and menus

More on ArcPy



- PDF with more explanation of ArcPY under our class file
 - Canvas -> Files -> Lectures -> More on ArcPY.pdf

Database Design



- Objectives:
 1. Eliminate Data Redundancy
 2. Ensure Data Integrity and Accuracy