22 Spatial Joins in a Can:

Combining ArcGIS and Python Data Structures for Data Formatting

Andy Bradford
City of Abilene, Texas

Replacing MAR

- Master Address Repository, from 2010:
  - Prepared from geodatabase
  - Individual address changes entered with Workflow Manager Extension

Used for:

- Water Billing
- Building Inspections
- Code Enforcement
- Stormwater
- Animal Control
- Elections
- Stormwater
- Planning
- Community Enhancement

Replacing MAR

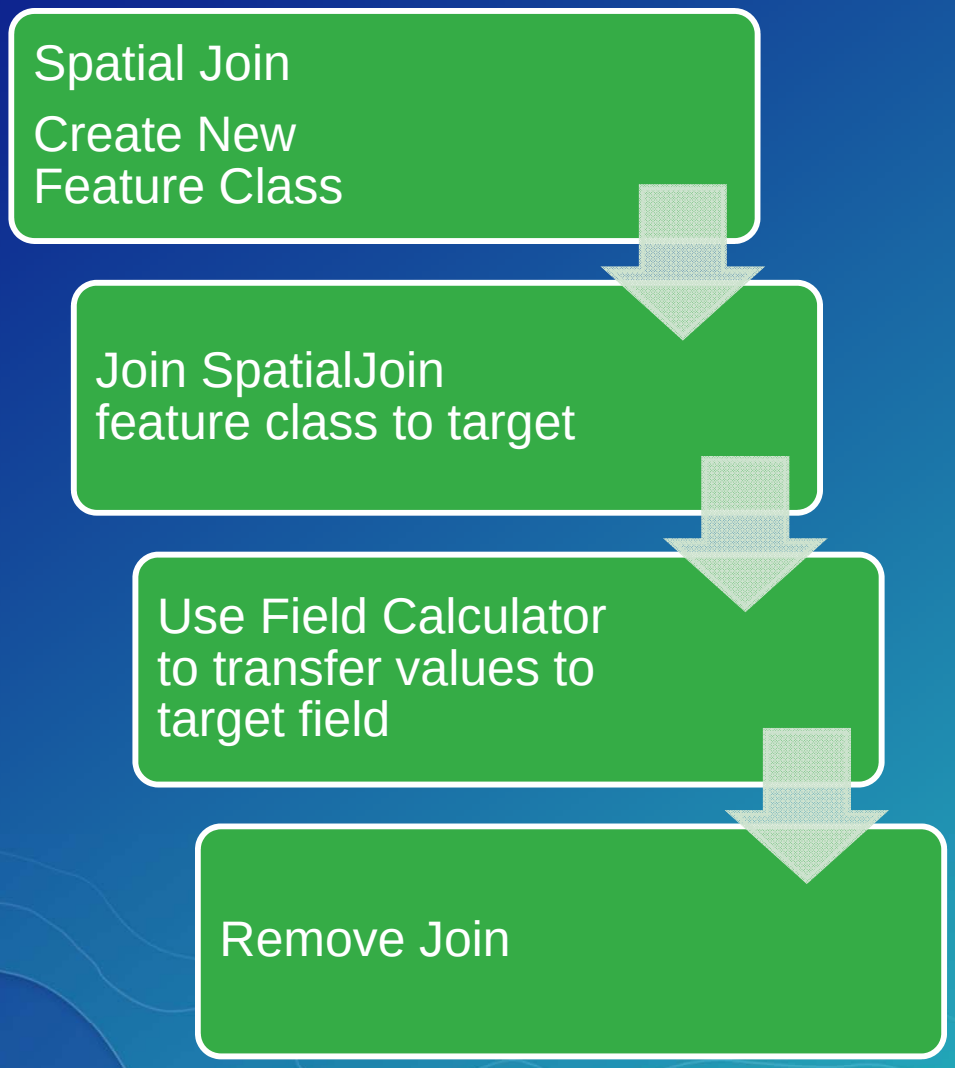
Replacement would have to incorporate:

- Parcel Boundaries
- Address Points – possibly multiple per parcel
- Parcel Ownership & Tax Information
- Parcel Spatial Information – 22 fields
  - School district & Attendance Zones
  - Incorporated City
  - Land Use
  - Zoning
  - Flood Zone
  - Acreage

Courier

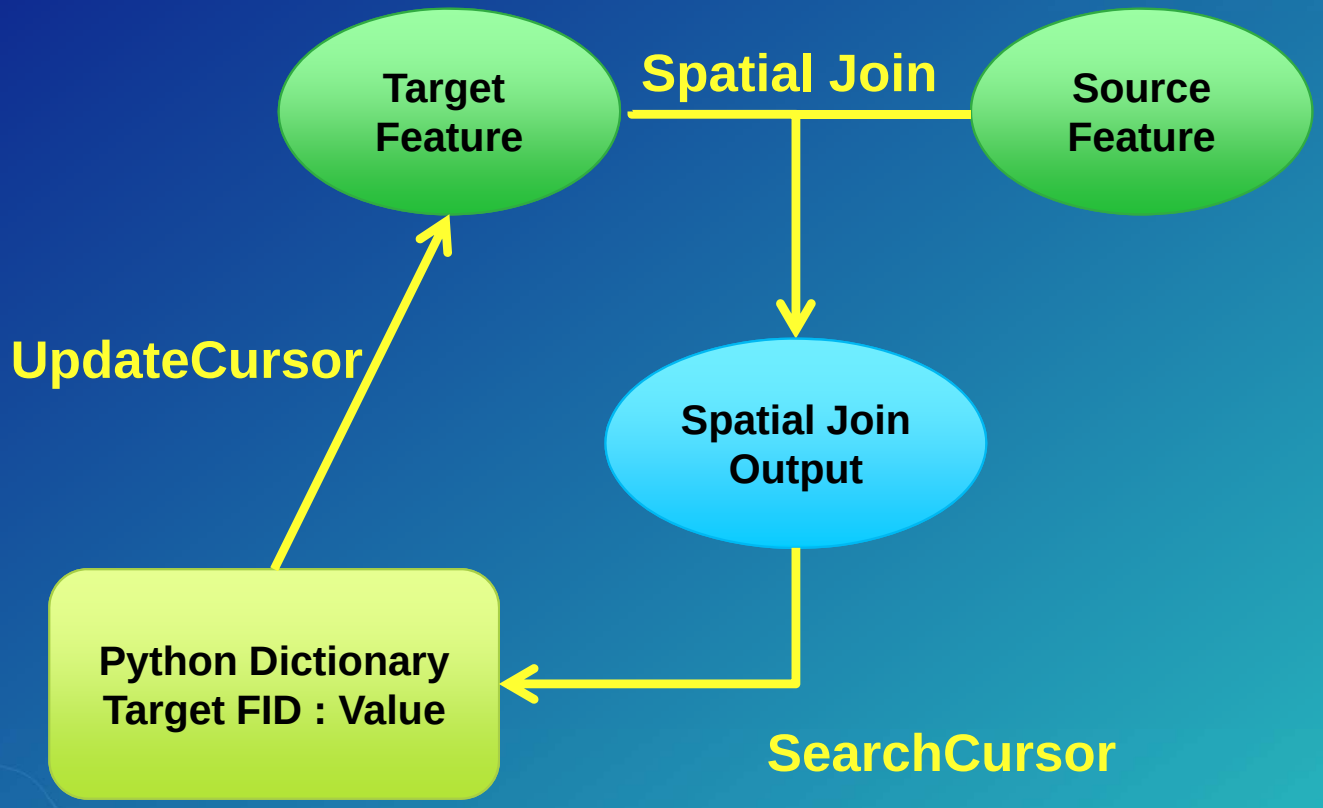
- Daily Python Script to create address table
- Spatial Join of Parcels and Address Points
  - One-to-many join creates duplicate parcel for each address
- Table Join to gather ownership/tax information
- But what about the 22 other fields?
- And fields for future needs?

Pure Arc



- Again and again, 24 Times.
  - Plus undetermined future fields!
- And what if field names are the same between source and target?

Python Cuts The Corners



Python Structures – Spatial Field Retrieval

- Spatial Join to a scratch file with a single target field
- Use SearchCursor to scan target field into a dictionary, with target FID as key
- Use UpdateCursor to put values in target features with corresponding FID

```
#Create field map a la forrestchev
#this field mapping code sets up the Spatial Join code later
#to create an output with only the Target_FID and the source field.
ScratchFMS = arcpy.FieldMappings()
ScratchFMS.addTable(SourceLayer)
SourceIndex = ScratchFMS.findFieldMapIndex(SourceField)
SourceFM = ScratchFMS.getFieldMap(SourceIndex)
ScratchFMS = arcpy.FieldMappings()
SourceFM.addInputField(SourceLayer, SourceField)
SourceFM.mergeRule = MergeRule
ScratchFMS.addFieldMap(SourceFM)
```

```
#spatial join to scratch features
arcpy.SpatialJoin_analysis(InLayer, SourceLayer, "ScratchSJ", "JOIN_ONE_TO_ONE",
                           "KEEP_ALL", ScratchFMS, SpatShip, SearchDist)
arcpy.AddMessage("Spatial Join completed.")

#create dictionary object for join purposes.
#the key will be the Target FID, and the value is the target field value.
JoinDict = {}
with arcpy.da.SearchCursor("ScratchSJ", ("TARGET_FID", SourceField)) as cursor:
    for row in cursor:
        fid = row[0]
        val = row[1]
        #fix for issue that doubles feature counts
        if MergeRule == "COUNT":
            val = val/2
        JoinDict[fid] = val
```

```
#Update cursor, hinges on dictionary
with arcpy.da.UpdateCursor(InLayer, ("OID@", InField)) as cursor:
    #reach into dictionary using FID values
    for row in cursor:
        #Search for dictionary item with feature's FID as key
        val = JoinDict[row[0]]
        #fix for bug that throws exception if null values.
        #this leaves previous, presumably valid, value in that record.
        if val <> None:
            row[1] = val
        cursor.updateRow(row)
```

How To Retrieve 24 Fields? Combine Arc and Python Structures

- Geodatabase Table scanned by SearchCursor
  - Each row is a single field's information
  - Target field, source file, source field, field type
  - Create the field if it's not in the Courier file
  - Then performs a spatial field retrieval for that field
- Need to add a field to the list? Simply add record to control table

OBJECTID *	NAME	TYPE	LENGTH	PREC	SOURCE	LYR	SCFLD
2	ZONING	TEXT	10		\\gis\data\AddressStage.gdb\ZoningDistrict	ZONE	ZONECLASS
3	VOTER_PCT	DOUBLE	19	11	\\gis\data\AddressStage.gdb\VOTER2011	VOTER	VOTER_PCT
4	COMM_PCT	DOUBLE	19	11	\\gis\data\AddressStage.gdb\VOTER2011	VOTER	COMM_PCT
5	JP_PCT	DOUBLE	19	11	\\gis\data\AddressStage.gdb\VOTER2011	VOTER	JP_PCT

Field Name

Field Attributes

Field Source File

Source Field Name

Courier

Prep
- Import files from SDE to staging GDB
- Rename or Create Fields as needed

Spatial Retrieval
- Run on list table held in GDB
- Populates 22 fields

Parse
- Parse address fields from SDE to Courier format
- Single UpdateCursor

Table Join
- Run on list table held in GDB
- Retrieves ownership and land value information

Export
- Export table to destination database
- Export completed parcels back to SDE for publishing

“Pigeon” – Same Concept But Bigger

- Feed Street/Address data to Computer Aided Dispatch System
- Multiple feature classes:
  - Streets
  - Addresses
  - Common Place Names
  - Zones
- Create and Publish Map Services
- Create and Publish Locator Services


Thank you!
Andy Bradford
Email: Andy.Bradford@abilenetx.com
Github: mapping-glory