Chapter 1

Designing the geodatabase schema

One of the most important things you can start with in creating a geodatabase is a good design—and that design starts with a well-thought-out process about what data the geodatabase will house, how it will be edited, and how it will be maintained over time. Of course, a geodatabase will contain features classes, but whether to include a feature dataset, field names, domains, subtypes, and other functionalities should all be considered, and designed, before your fingers ever touch the keyboard. Case in point, this entire first chapter can be worked strictly with paper and pencil. These elements are all complex and cannot be created on the fly if you hope to achieve maximum success. Think about each of the components, examine what type of data will be used, imagine yourself or others having to edit this data, or even think about how you might use the data in a geoprocessing task. Perhaps you've used a poorly designed dataset before and spent a large amount of time formatting the data just to make it usable. If you can remember what problems arose and what steps you took to correct them, you will have a good understanding of how to create your datasets and avoid those mistakes.

It may also help to imagine what a final output map containing this data might look like so that you can determine what fields to use for labeling and symbolizing. The field types you design will make calculations either very easy or very difficult. Labeling will either be a snap or a headache. All these things can affect the design of a geodatabase, and a good design will make all future work that much easier.

Tutorial 1-1: Creating a geodatabase–building a logical model

The newest release of ArcGIS® Desktop software includes many additions and improvements to the data storage capabilities of the Esri® geodatabase file structure. There are new techniques to control interactions with the data, assign a behavior to it, and define relationships among datasets. In the design process, it is important to understand these techniques to build the most efficient database possible. It is even possible to build data controls to aid the user in creating data while decreasing the chance of introducing errors into the dataset. This book was tested using ArcGIS Pro 2.3.

LEARNING OBJECTIVES

- Outline geodatabase behavior
- Integrate datasets
- Model reality
- Use ArcGIS® Pro

Introduction

The goal of designing a geodatabase is to model the reality it is intended to represent. There are many characteristics, or behaviors, of the data that can be included in a geodatabase using various techniques. As the data modeler, it is your job to explore the capabilities of ArcGIS to make the most efficient and flexible database possible. The time spent at the start of a project designing the geodatabase will reap rewards later by making the data easier to use and edit and presenting a better representation of reality.

The first step is to study the reality that is about to be modeled. Look carefully and determine what features must be included in the geodatabase. In ArcGIS, everything is modeled as points, lines, and polygons, so realistic characteristics will need to be assigned to these pieces.

Next, look at how the data will be created. Will it be imported from another source, collected with field equipment, traced from aerial photos, drawn from survey data, or derived from some other process?

Finally, consider how this data will be used. Who will perform the edits, and what queries might it be expected to support in the future? Knowing many of the questions that the data will be used to answer will shape the geodatabase design.

With these things in mind, you can start to construct a logical data model. The model will diagram your process and allow for updates and changes before the final design is committed to a geodatabase. The design process uses a spreadsheet-based form to enter all the characteristics of the geodatabase, the feature datasets, the feature classes, and the data integrity rules of the feature classes such as domains and subtypes. Microsoft Excel® spreadsheet files included with the exercise data follow the creation process in ArcGIS Pro that will be used for basic diagramming.

The logical model is used to show what data types you will have (points, lines, or polygons), what tabular data will be included in the dataset, and the relationships between the tabular data and feature classes (if any). The model is also easily shared among your colleagues, so you can get several opinions on the design you are attempting.

Once a preliminary logical model is completed, it can be checked against many of the advanced features of the geodatabase, such as domains, subtypes, and relationships. Have the best tools been employed to ensure data integrity, ease of editing, and future expansion? You will also check the geodatabase against your idea of how the data should behave.

The result should be a well-thought-out geodatabase that is both efficient and a good representation of reality . . . well, at least as close as you can get using points, lines, and polygons.

You will start with a simple geodatabase, and then examine several advanced geodatabase options to see if better efficiency and more realistic behaviors can be achieved. The good news about any geodatabase design is that if it works, it's a success. Ten different people could design 10 different geodatabases for the same project, and they all could work quite well. The true test is how efficient a geodatabase is, how well it models reality, how well it maintains data integrity, how flexible it remains for future projects, and how easy it is to work with in editing and extracting information.

Designing the data

Scenario

The City of Oleander, Texas, population 60,000, has hired you as the top-gun geodatabase designer and wants an all-new database design for its parcel data. The new geodatabase will be used to designate each piece of property in the city—who owns it, its legal description, address, and more. You'll get information from the city planner to get the full idea of what's needed, and then create a diagram of your proposal. At this point, the geodatabase will not be constructed, only designed.

The city planner describes a dataset that would have a polygon for each piece of property in the city, whether it is platted or unplatted. It should have information about the legal description, the street address, and the current usage of each property.

Data

Because you are creating this geodatabase from scratch, there is no data to start with. You will need to print the geodatabase design forms from the exercise materials you download from ArcGIS® Online and use them to document the design process.

Tools used

- Geodatabase design forms

Begin the logical design for the geodatabase

The main component of this geodatabase will be polygons representing every piece of property in the city. Each piece of property is assigned certain data by the city. This data includes the subdivision name, block designation, lot designation, street address, and a land-use code, which shows how the land is being used.

As you know, the geodatabase is the framework in which other components are built. It may contain feature classes, tables, relationship classes, feature datasets, and many other

components. You will design the geodatabase and its components using the geodatabase design forms provided in the files you download for this book.

Access the data

The data is stored in a book group named Focus on GDBs in ArcGIS Pro (Esri Press) in the Learn ArcGIS organization. You will log in and access the files for this tutorial, as well as for all the tutorials and corresponding exercises in this book as you need them.

1. **Go to https://www.ArcGIS.com, and log in with an ArcGIS Online account.**

2. **On the Home tab, type** Focus on GDBs in ArcGIS Pro **in the Search box, and then click the Search for Groups entry in the drop-down list. (If no groups were found, turn off the option to "Only search in your organization.")**

3. **Click the link to open the Focus on GDBs in ArcGIS Pro (Esri Press) group and find the data, named FocusGDB. It consists of a zip file for each tutorial in the book.**

4. **Create a folder named** EsriPress **in a location where you want to store all the files, preferably on your drive C. Click the thumbnail and download the data for the first tutorial, saving it to your new folder (e.g., C:\EsriPress), rather than in the Documents library or on the desktop.**

5. **Extract the zip file. It will create a folder named Tutorial 1-1.**

6. **You can download and access these files as you need them for the other tutorials in this book.**

Start building the model

1. **Open a file explorer window and navigate to the location where you downloaded the tutorial 1-1 materials. Open the file GDB design forms.xlsx. Note: There is also a PDF version of this file in the same location.**

2. **Print all six pages of the geodatabase design forms (GDB feature classes, tables, domains, domains 2, subtypes, and relationships).**

   You will write out all your designs on these printed sheets, and you can print more sheets for corrections or expanded designs. Pages 1 and 2 will be used for these first few steps, but the other pages will be used in the other steps of this tutorial.

The geodatabase will need a name. It should reflect in general terms what will be stored in it.

3. **On the first line of page 1 (GDB feature classes), write the name** LandRecords **for the geodatabase name.**

The next line asks for a feature dataset name. A feature dataset is used to separate data into smaller subsets but is also important in grouping data for use in topology and various other advanced features. For now, leave the feature dataset name blank.

The next step is to start filling in the feature classes. So far, the city planner has described only one feature class, which will contain the polygons representing parcels and will include the fields he described.

4. **On the feature class portion of the worksheet, add a new feature class named** Parcels. **Note its type as POLY (for Polygon), and give it an alias of** Property Ownership.

Geodatabase design forms		*ArcGIS Pro*
Geodatabase name		**LandRecords**
Feature dataset name		
Feature classes:		
Type	Feature class name	Alias
POLY	**Parcels**	**Property Ownership**
	Type:	Indicate if this is a **PoiNT**, Line, or **POLY**gon feature class.
	Name:	Enter the name of your feature class.
	Alias:	Describe the contents of the feature class.

The alias is one of the first characteristics of the geodatabase that will be assigned. This alias will be shown in the Contents pane when the layer is added to a project; consequently, it also could be used in a legend. The alias should be very descriptive of the data to distinguish it from other datasets.

This feature class will have fields to store data, and these fields are recorded on the second design form. From the city planner's description, you can determine that the feature class will have fields for the subdivision name, block designation, lot designation, street address, and a land-use code. The content of the first three fields is self-explanatory. They will need to contain alphanumeric characters, so their field types will be Text.

5. **On the Tables worksheet (page 2), write the name of the new feature class on the first line. Under the field name, enter the first field as** Sub_Name**. Note its type as** Text**. Add another field for** Blk **as** Text **and** Lot_No **as** Text**.**

Tables worksheet							
Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
Parcels	Sub_Name		Text				
	Blk		Text				
	Lot_No		Text				

Simple so far, but there is other information to enter that will start impacting the future use of the data. The first is the field alias, which is another characteristic of the geodatabase. The alias is typically similar to the field name, with the important difference that it is allowed to have spaces in the text. This alias will be shown in many of the ArcGIS tools, the attribute table, any classification schemes, and many more places when the data is accessed. The field alias also should be descriptive of what data the field contains.

6. **Next to the field name Sub_Name, write the description** Subdivision Name **as the alias. Then write the alias** Block Designation **for Blk and** Lot Number **for Lot_No.**

Tables worksheet							
Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
Parcels	Sub_Name	Subdivision Name	Text				
	Blk	Block Designation	Text				
	Lot_No	Lot Number	Text				

Those notations have taken care of some of the fields, but there are more. The next is street address information. The address could be entered as a single field, but if you ever want to geocode against this dataset, it would be better to have each component of the address in a separate field. The common fields for geocoding are street prefix type, prefix direction, address number, street name, street type, suffix direction, and zip code. Fields such as city name or state name may be necessary if you are geocoding a broader region, but because all the features that this dataset will contain are specific to Oleander, you can leave them out. All the listed fields must be included in the table.

One interesting thing is that if the fields are given certain names that ArcGIS uses as a standard for address components, they will be filled in automatically when you make an address locator. An address locator is a special file that ArcGIS builds using your dataset that will allow addresses to be found easily when geocoding or using the Find tool. This address locator can also be used in routing and network applications. A list of preferred field names for each field is stored in the address locator style file, which was loaded when you installed ArcGIS Pro. You can open the file in the <install directory>\ArcGIS\Pro\Resources\Locators

folder and view the list by searching for the phrase *Preferred Field Names*. You may add your own field names to the list as needed.

7. **Open a file explorer window. Navigate to the folder containing your ArcGIS installation (e.g., C:\Program Files\ArcGIS\Pro\Resources), and open the Locators folder. Scroll down to the file USAddress.lot.xml, right-click, and click Open with > Notepad.**

8. **Use the Find tool, or scroll down to the area labeled "Reference data style for Single House."**

```
<!--This section defines the reference data styles, including reference data tables, fields, -->
<!--preferred field names, reference to the mapping schema, and join clauses for the primary -->
<!--and alternate name tables if it applies. The name of the style is displayed on ArcGIS -->
<!--geocoding Select Address Locator dialog box.-->
<ref_data_styles>
  <!--Reference data style for Single House (Address Points)-->
  <ref_data_style>
    <name>Single House</name>
    <type>Address</type>
    <priority>6</priority>
    <desc>US Single House Addresses</desc>
```

9. **Within the section, scroll down to the area labeled Primary.House.**

```
<field_role name="Primary.House" required="true">
  <display_name>House Number</display_name>
  <preferred_name>PremiseNumber</preferred_name>
  <preferred_name>HOUSE</preferred_name>
  <preferred_name>HOUSENUM</preferred_name>
  <preferred_name>HOUSE_NUM</preferred_name>
  <preferred_name>HOUSE-NUM</preferred_name>
  <preferred_name>HOUSE_NUMBER</preferred_name>
  <preferred_name>HOUSE-NUMBER</preferred_name>
  <preferred_name>HOUSE_NUMB</preferred_name>
  <preferred_name>HOUSE-NUMB</preferred_name>
  <preferred_name>HOUSENO</preferred_name>
  <preferred_name>HN</preferred_name>
  <preferred_name>NUMBER</preferred_name>
  <preferred_name>ADDNUMBER</preferred_name>
  <preferred_name>ADD_NUMBER</preferred_name>
  <preferred_name>ADD</preferred_name>
  <preferred_name>ADD_HN</preferred_name>
  <preferred_name>ADDRESS</preferred_name>
  <preferred_name>ADDRESS_NUM</preferred_name>
  <preferred_name>ADDR</preferred_name>
  <preferred_name>ADDR_HN</preferred_name>
  <preferred_name>SITENUMBER</preferred_name>
  <preferred_name>SITE_HOUSENUM</preferred_name>
  <preferred_name>SITUS_NUM</preferred_name>
  <preferred_name>STNUMBER</preferred_name>
  <preferred_name>ST_NUMBER</preferred_name>
  <preferred_name>STR_NUMBER</preferred_name>
  <preferred_name>STREET_NUM</preferred_name>
  <preferred_name>CIVICNUMBER</preferred_name>
</field_role>
```

As you can see, there are many acceptable names for this field. The advantage of using one of these suggested field names over a new incarnation is that prebuild geocoders and geoprocessing tools will automatically recognize these field names and, in some cases, can validate that you have the correct field name chosen for a given data type. Look over the rest of the list to see what the choices are for other field types such as Primary.StreetName, Primary .Locality, or Primary.Postal. It is important to try to use one of the preset abbreviations or add your own field names to this list whenever possible. This use of existing names will make the entry of field names in certain tool parameters almost automatic. The ones shown in this geodatabase design were all derived from this list.

10. **On the Tables worksheet, write the following field names, field types, and aliases:**
    - Pre_Type, Text, Prefix Type
    - Pre_Dir, Text, Prefix Direction
    - House_Num, Text, House Number
    - Street_Name, Text, Street Name
    - Street_Type, Text, Street Type
    - Suffix_Dir, Text, Suffix Direction
    - ZIP_Code, LI (Long Integer), ZIP Code

These fields will add a lot of functionality to the dataset that may be valuable later. For example, you could select all the parcels in a certain subdivision, all the parcels that front a certain street, or use the House_Num field to put address labels on the map.

The last bit of data that the city planner mentioned was the land-use code. The land-use code is split into two sets of codes. The primary land-use code is one of seven main codes, and each primary code has many secondary codes that are more descriptive of the land use. The use of two codes helps in making both a generalized land-use map and a detailed land-use map.

11. **On the Tables worksheet, add the field** Primary_Use**, with the field type** Text **and the alias** Primary Land Use Code**, and the field** Secondary_Code**, with the field type** Text **and the alias** Secondary Land Use Code**.**

The data entered so far has involved information that the city planner wanted. One more piece of data is necessary for you to maintain a connection to certain third-party data that is important to the project. The identity of the property owner is not stored in the parcel's attribute table but is stored in an external table. You will need to add a field to your data structure that will allow you to set up a relationship between the field and the external table. The procedure is discussed later in the tutorial, but for now, you will need to add a field to accommodate the relationship.

12. **On the Tables worksheet, add the field** Georeference**, with the field type** Text **and the alias** Georeference Index**.**

Tables worksheet							
Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
Parcels	Sub_Name	Subdivision Name	Text				
	Blk	Block Designation	Text				
	Lot_No	Lot Number	Text				
	Pre_Type	Prefix Type	Text				
	Pre_Dir	Prefix Direction	Text				
	House_Num	House Number	Text				
	Street_Name	Street Name	Text				
	Street_Type	Street Type	Text				
	Suffix_Dir	Suffix Direction	Text				
	ZIP_Code	ZIP Code	LI				
	Primary_Use	Primary Land Use Code	Text				
	Secondary_Code	Secondary Land Use Code	Text				
	Georeference	Georeference Index	Text				

Design for data integrity

The design looks good so far, but imagine what will happen when people start putting data in the table. If they left the Sub_Name field blank, there would be no way to identify the legal record of a piece of property. What about address number or the land-use code? These fields shouldn't be left blank, or there could be gaps in the data. On the other hand, not every street will have a value for prefix type, so there will be instances when a field value can be left blank and still be correct. You can set up parameters in the geodatabase to control these data integrity rules.

One way to build data integrity rules into your table is to set the flag for allowing null values, or no value, for a field. If nulls are not allowed, a validation check of the data will produce an error for any records entered without all the necessary values being provided. Perhaps, as an example, the person entering the data accidentally skipped the field during data entry or tried to enter data before all the information was known. Either way, it could cause problems with your data if nulls are not allowed.

The solution is to mark in the design table which fields are allowed to have nulls and which must have a value entered.

1. **On the Tables worksheet, mark the following fields to allow null values by placing a *Y* in the Nulls column:**
   - **Pre_Type**
   - **Pre_Dir**
   - **Suffix_Dir**

2. **Mark the remaining fields as not allowing null values by placing an *N* in the Nulls column:**
   - **Sub_Name**
   - **Blk**
   - **Lot_No**
   - **House_Num**
   - **Street_Name**
   - **Street_Type**
   - **ZIP_Code**
   - **Primary_Use**
   - **Secondary_Code**
   - **Georeference**

Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)
Parcels	Sub_Name	Subdivision Name	Text	N
	Blk	Block Designation	Text	N
	Lot_No	Lot Number	Text	N
	Pre_Type	Prefix Type	Text	Y
	Pre_Dir	Prefix Direction	Text	Y
	House_Num	House Number	Text	N
	Street_Name	Street Name	Text	N
	Street_Type	Street Type	Text	N
	Suffix_Dir	Suffix Direction	Text	Y
	ZIP_Code	ZIP Code	LI	N
	Primary_Use	Primary Land Use Code	Text	N
	Secondary_Code	Secondary Land Use Code	Text	N
	Georeference	Georeference Index	Text	N

Another data integrity component is the domain. A domain allows you to define a list of values for any text field or a range of values for a numeric field. When data is entered, it is matched against the domain to see if it is a valid value. This designation helps eliminate typos or inventive abbreviations. Imagine 10 data entry clerks all coming up with unique abbreviations for the land-use code Vacant. It might be entered as VAC, V, Vcnt, or any number of misspellings. A query to find all vacant property would be difficult. If a domain is applied to the field Primary_Use that contains only the seven correct category abbreviations, it would be impossible for anyone to enter a value that wasn't in the domain.

In addition to the defined primary-use codes, there are secondary-use codes associated with each primary-use code. For instance, the primary code Commercial has a set of secondary codes named Light Commercial, Special District, Church, and School. Using a contingent value domain, you can restrict the field selections for a second field on the basis of the first

field's value. The Secondary Land Use Code field will get a domain with all the available land-use descriptions in it, but later you will build a matrix that will pair the Primary Land Use Code with only those values in the domain that are relevant to the selected code. This pairing will prevent the user from selecting a secondary-use code that does not match the category of the primary-use code.

The domain values will be entered on the Domains worksheet, and you will note in the worksheet that it is a domain to avoid confusion with subtype fields that may be entered later.

3. **On the Tables worksheet, add an entry on the line for Primary_Use with the name of a domain that will contain the acceptable values for this field. Call it** Prim_Use_Codes**, and place a** (D1) **in front of it for "domain number 1."**

4. **On the line for Secondary_Code, add the notation** (D2) **and the name** Sec_Use_Code.

Primary_Use	Primary Land Use Code	Text	N	(D1) Prim_Use_Codes
Secondary_Code	Secondary Land Use Code	Text	N	(D2) Sec_Use_Code

5. **Now turn to the Domains worksheet (page 3), and write the domain name** Prim_Use_Codes**, with a description of** Primary Use Codes for Parcels**, a field type of** Text**, and the type of domain as** Coded Values.

6. **In the Code column, write** VAC**, with a description in the Value column of** Vacant Property**. Under that, write** RES**, with a description of** Residential Property**. Continue down the form, entering the rest of the Prim_Use_Codes values from the accompanying list. Print more worksheets if necessary.**

   COM Commercial Property
   IND Industrial Property
   GOV Government Property
   PRK Park Land
   OTHER Other Uses

Domains worksheet

	Domain name	Description	Field type	Domain type	Coded values / Range Code (Min)	Value (Max)
D1	Prim_Use_Codes	Primary use Codes for Parcels	Text	Coded Values	VAC	Vacant Property
					RES	Residential Property
					COM	Commercial Property
					IND	Industrial Property
					GOV	Government Property
					PRK	Park Land
					OTHER	Other Uses

Adding this domain will build a validation check for data integrity. You can rest assured that the primary-use code abbreviation entered for any piece of property will fit your normal list. But one concern might be that someone could set the primary-use code to Commercial, and then set the secondary-use code to 35th3. These values don't match, so a domain for the secondary codes should be designed to include all the detailed codes. In the form, a note can be added to show which primary land-use code each value will pair with, and later you will see how these codes are paired to further constrain the values entered.

7. **On the Domains worksheet, add a new domain name (D2) of** Sec_Use_Code**. Give it a description of** Secondary Land Use Codes**, with a field type of** Text **and a domain type of** Coded Values**. The codes that will go in this domain are as shown in the figure. Print more copies of the worksheet as needed.**

Coded values / Range		
Code	(Min)	Value (Max)
VAC	PRIV	Vacant Private Land
VAC	GOV	Vacant Government Land
RES	SF_DET	Single Family Detached
RES	SF_LIM	Single Family Limited
RES	DUP	Duplex
RES	TRI	Triplex
RES	QUAD	Quadruplex
RES	CONDO	Condominiums
RES	MANUF	Manufactured Housing
RES	TOWN	Townhomes
RES	MULTI	Multi-Family
COM	LT_COM	Light Commercial
COM	SPEC	Special District
COM	SCH	School
COM	CHR	Church
IND	LT_IND	Light Industrial
IND	HVY_IND	Heavy Industrial
GOV	CITY	City Owned
GOV	CNTY	County Owned
GOV	STATE	State Owned
GOV	US	Federally Owned
PRK	PUB	Public Park
PRK	PRIVPRK	Private Open Space
OTHER	ROW	Right-of-way
OTHER	PROW	Private Right-of-way
OTHER	UTIL	Utility
OTHER	ESMT	Easement

Perhaps there are other fields in the table that would benefit from the application of a domain. Most of them, however, such as a subdivision name or a house number, couldn't be constrained in this way; there would be too many values. But the field street type might be a good candidate. The US Postal Service has a standard set of street type abbreviations, and from time to time, you may be asked to generate a mailing list from this table. So, it would be a good idea to add a domain to this field.

The table includes many acceptable street type abbreviations; you would not want to list all of them on the Domains form, and you wouldn't want to type them into a domain. So, a command exists to take a file listing of street types and read them into a domain, and a file named Suffix.txt with these abbreviations is provided in the downloaded materials. The abbreviations it contains were found on a US Postal Service website, and the file contains all the recognized suffix names. The process of turning this file into a domain will be demonstrated in tutorial 2-1, but for now you can write the file name Suffix.txt on the design worksheet.

8. **On the Tables worksheet, write** St_Type_Abbrv **as the domain name for the field Street_Type, and add** (D3)**, noting that it is a domain.**

9. **Next, go to the Domains worksheet, and write the name** St_Type_Abbrv**, a description of** Street Type Abbreviations**, a field type of** Text**, and a domain type of** Coded Values**. Under Code, write the file name to identify the file holding the domain values.**

D3	St_Type_Abbrv	Street Type Abbreviations	Text	Coded Values	Tutorial 1-1/Suffix.txt

Now is a good time to investigate other aspects of how the data will be used and see if there are any other data integrity techniques that might be employed—most notably, the subtypes.

Consider the situation with property. It is either platted by a legal survey or unplatted and recorded as a single deed. You might separate property as either platted (divided into developed lots with utilities) or unplatted (raw agricultural land). It is important to know the distinction for legal purposes and for the sale of property. It would be possible to put the platted land into one feature class and the unplatted property in another. If both feature classes were in the same geodatabase, they could both be easily stored and edited at the same time. The symbology and annotation would work well for both, and each feature class could have different data integrity rules. So dividing them into two feature classes would work and might be beneficial.

But consider how the data might be used in a query. If a list of all property owners in a given region was needed, it would have to come from two separate files, and exporting the list would create two tables. Although it would be beneficial in some respects to put property data into two feature classes, using the data would be problematic. That's where subtypes can be valuable.

Using a subtype is a way to create a virtual subdivision of data within the same feature class, and then apply different data integrity rules to each category. It's the best of both worlds: the data can be separated into logical categories and be given data integrity rules for each category but keep the convenience of being edited, queried, and managed in

a single feature class. You'll also see later, in tutorial 1-2, how subtypes can be used to set default values, establish unique attribute domains, set connectivity rules, and establish relationship rules for each subcategory created. They'll even make it easier to symbolize and label data.

A field to contain the subtype code must be added to the table. The field type must be Integer, and the codes will be established along with a description. For this data, you'll make a code 1 for Platted Property, code 2 for Unplatted Property, and code 3 for Plat Pending. This last code will be for property that has been approved by the city but is awaiting the filing data from the county. This will be a simple subtype, without any additional data integrity rules added.

10. **On the Tables worksheet, add a new field on the bottom named** Plat_Status**, make its field type** SI **(short integer), give it an alias of** Plat Status**, and don't allow for null values. Because most new property being added to the dataset will be platted, record its default value as** 1**. Finally, write the name** Plat_Subtype **for the subtype name, with a notation of** (S1)**.**

Plat_Status	Plat Status	SI	N	(S1) Plat_Subtype

11. **Go to the Subtypes worksheet (page 5). Write the name of the subtype as** Plat_Subtype**, and add the three codes described previously:**
    1 = Platted Property
    2 = Unplatted Property
    3 = Plat Pending

Subtypes worksheet						
				PRESET DEFAULTS		
	Subtype name	Code	Description	Field	Domain name	Default value
S1	Plat_Subtype	1	Platted Property			
		2	Unplatted Property			
		3	Plat Pending			

Extend the data model

This work concludes the initial design phase of the Parcels feature class, but there's another component to investigate. When these polygons are symbolized, they can each have a solid fill and a line style for their perimeter. When maps are made, however, the boundaries of the parcels must be symbolized differently. The edge of the parcel that fronts a street will be drawn with a thicker line; the edges representing property lines between properties will be a thinner line; if someone owns two adjacent pieces of property, the line between them should be dashed.

Consider creating a set of lines that will duplicate the boundaries of each parcel. Then these lines can be symbolized as described. The only field the feature class will need is a code describing which type of line to draw. This field would benefit from having a data integrity rule (a domain) with the three categories of lines described.

A behavior will need to be created between the polygons representing property and the lines representing their boundaries. If the shape of any polygon is modified, the lines will need to automatically adjust to coincide. This type of relationship is called *topology* and will be discussed in chapter 7. For ArcGIS to manage this topology, the feature classes must reside within the same feature dataset.

Feature datasets are another way to segregate data inside a geodatabase. If any behavior is to be built for a feature class, such as topologies, network databases, geometric networks, relationships, or terrains, the feature class must reside in a feature dataset. For this example, you will establish a feature dataset for your feature classes, so that the corresponding topology can be built.

1. **On the Geodatabase worksheet (page 1), write the name of the feature dataset as** PropertyData. **Next, write the new feature class name** LotBoundaries **on a blank line. Give it a feature type of** LINE **and an alias of** Lot Boundaries.

Geodatabase design forms	*ArcGIS Pro*	
Geodatabase name	**LandRecords**	
Feature dataset name	**PropertyData**	
Feature classes:		
Type	Feature class name	Alias
POLY	**Parcels**	**Property Ownership**
LINE	**LotBoundaries**	**Lot Boundaries**

2. **Next, go to the Tables worksheet, and write the name of the new table as** LotBoundaries. **Then write the single attribute of this table,** Line_Code. **Give it a field type of** Text, **add an alias of** Line Code, **and do not allow nulls. Add a notation that there is a domain for this field, and name it** (D4) Parcel_Line_Codes.

LotBoundaries	**Line_Code**	**Line Code**	**Text**	**N**	**(D4) Parcel_Line_Codes**

3. **Finish by filling in the information for the domain. On the Domains worksheet (page 3), add the name of the domain as** Parcel_Line_Codes**, a description of** Line Codes for Parcels**, and a field type of** Text**, and note the domain type as** Coded Values**. Then write the three domain values described previously:**
   - ROW = Edge of Right-of-way
   - LOT = Lot Line
   - SPLIT = Split Lot Line

D4	Parcel_Line_Codes	Line Codes for Parcels	Text	Coded Values	ROW	Edge of Right-of-way
					LOT	Lot Line
					SPLIT	Split Lot Line

Design a relationship class

The features you've dealt with in the design so far have been the points, lines, and polygons that will create the model of reality. Not all the data you will need for this model, however, is in the form of points, lines, and polygons. The design will also need to include tabular data that is provided by an outside agency. For each parcel, a county appraisal agency provides ownership and value information. This data would be valuable for analysis if it was associated with the parcel data. The nature of the table is that it is updated regularly from separate appraisal software, so it cannot be incorporated in the polygon feature class in the same way as regular data. By keeping it separate, it will facilitate the maintenance of both ArcGIS use of the data and the third-party software's use of the data.

A relationship class has many of the benefits of a simple join in a project but also provides a mechanism for controlling edits in the related table. If the graphic features are altered in editing, rules in the relationship class can also alter the related table and maintain the relationship. For this example, the parcels have a match in the appraisal roll table. If a piece of property is removed because of replatting, the associated record in the appraisal table can be set to be deleted automatically.

The final consideration is the cardinality of the relationship. If each parcel has one and only one match in the appraisal table, and vice versa, the cardinality is said to be one to one (1:1). If one parcel can have several matches in the appraisal table, such as the case of a single parcel being owned by more than one person, the cardinality is said to be one to many (1:M). If the opposite relationship was also true—that is, an owner can also own several pieces of property—the relationship is said to be many to many (M:N).

Armed with this information, you can move to the worksheet on relationship classes and fill in the details.

Relationship worksheet					
Name of the relationship class:					
Origin table/feature class:					
Destination table/feature class:					
Relationship type:	Simple (peer to peer)		Composite		
Labels:					
Origin to destination:					
Destination to origin:					
Message propagation:		Forward	Backward	Both	None
Cardinality:	1-1	1-M	M-N		
Attributes:	No	Yes - Table name:			
				Add to the tables worksheet	
		Primary key field			Foreign key name
Origin table/feature class:				:	
Destination table/feature class:				:	

1. **Continuing in the same design spreadsheet, on the Relationships worksheet (page 6), write the origin table as** Parcels **and the destination table as** TaxRecords_2019**. Name the output relationship class** Ownership**.**

Relationship worksheet			
Origin table/feature class:	**Parcels**		
Destination table/feature class:	**TaxRecords_2019**		
Output relationship class name:	***Ownership***		

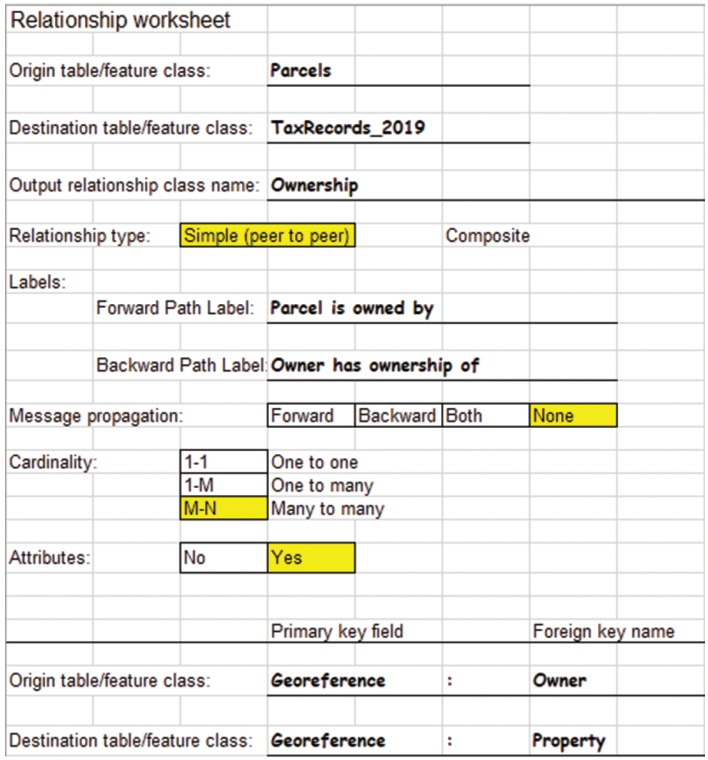
The relationship class can be used to add or delete records, but because the related table will be managed by another source, the relationship type should not allow records to be deleted, making it a simple (peer-to-peer) relationship. Labels will be shown to describe the

relationship between the tables. The description for moving from the parcels feature class to the appraisal table is "Parcel is owned by," and from the appraisal table to the parcels feature class, it is "Owner has ownership of." As the relationship is used in analysis, these labels will remind the user of the nature of the relationship. Normally, relationship classes are transparent to the user, but you can have your project controls display a message when the relationship is used. For this example, opt not to use them.

2. **Circle Simple (peer to peer) as the relationship type, and write the labels** Parcel is owned by **for Forward Path Label and** Owner has ownership of **for Backward Path Label. Circle None for Message propagation.**

   Next, you'll note the cardinality as many to many, since a parcel can be owned by several people, and one person may own several parcels. It may also be beneficial to store what percentage of ownership can be attributed to each owner. This notation will help when more than one person is recorded as the owner. You'll write the name of the table as Ownership_Rel, and it will be added to the Tables worksheet later. Finally, you'll select the fields that will be the basis for the relationship and give them a label describing their relationship to the related table (foreign key name).

3. **On your Relationship worksheet, circle M-N for Cardinality, and circle Yes under Attributes. Set the origin table and destination table primary key fields as** Georeference. **Name the origin table foreign key** Owner **and the destination table foreign key** Property.



Relationship worksheet

Origin table/feature class:		**Parcels**			
Destination table/feature class:		**TaxRecords_2019**			
Output relationship class name:		**Ownership**			
Relationship type:	Simple (peer to peer)			Composite	
Labels:					
	Forward Path Label:	**Parcel is owned by**			
	Backward Path Label:	**Owner has ownership of**			
Message propagation:		Forward	Backward	Both	None
Cardinality:	1-1	One to one			
	1-M	One to many			
	M-N	Many to many			
Attributes:	No	Yes			
		Primary key field			Foreign key name
Origin table/feature class:		**Georeference**	:		**Owner**
Destination table/feature class:		**Georeference**	:		**Property**

This work completes the logical model for the geodatabase. From these design forms, you will be able to create the entire structure and begin using it for storing data. If you do not have a lot of experience editing geodatabases, you may want to jump ahead to tutorial 2-1 and see how this design will function, and then come back to this exercise. Otherwise, complete this exercise, which will continue to focus on the design phase.

Exercise 1-1

The tutorial showed how to diagram a geodatabase to include feature classes along with their associated tables, domains, and subtypes. The goal was to think through the design, adding data integrity and behavior guidelines to the database.

In this exercise, you will repeat the process for another dataset required by the city planner. This one will contain the zoning data for Oleander. The zoning code for a piece of property determines the type of development that is allowed on a parcel (even though the land use may be different). The zoning districts may incorporate several parcels and generally follow parcel boundaries, but they can split parcels, too.

The zoning districts will be represented by solid shaded polygons, so you will want to design a polygon feature class for districts. The edges of the polygons should be symbolized in one of two ways—either as a solid line representing a zoning boundary or a dashed line representing a change in allowable development density. Because of this scenario, you will want to design an additional linear feature class for symbology purposes. The codes necessary for the zoning information are as follows:

R-1	**Single Family Residential**
R-1A	**Single Family Attached**
R-1L	**Single Family Limited**
R-2	**Duplex**
R-3	**Triplex**
R-4	**Quadruplex**
R-5	**Multifamily**
C-1	**Light Commercial**
C-2	**Heavy Commercial**
TH	**Townhomes**
LI	**Limited Industrial**
I-1	**Light Industrial**
I-2	**Heavy Industrial**
TX-121	**121 Development District**
POS	**Public Open Space**

Analyze the descriptions of this data and determine what feature datasets and feature classes must be made, what fields they should contain, any domains that might

need to be created (and possibly contingent domains), and any subtypes that might be beneficial.

- Print a set of the geodatabase design forms as necessary.
- Use the forms to create the logical model for feature classes for the zoning polygons and zoning boundaries.
- Investigate the use of domains and subtypes to build data integrity and behavior into your design.

WHAT TO TURN IN

If you are working in a classroom setting with an instructor, you may be required to submit the design forms you created in tutorial 1-1.

- The completed geodatabase worksheets for:
  - Tutorial 1-1
  - Exercise 1-1

Review

Over the last 30 years, the way that geographic features have been portrayed, stored, and manipulated in geographic information systems (GIS) has evolved from a file-based technology into the present-day Esri geodatabase format. By using the Esri geodatabase, GIS practitioners can more realistically manage geographic features and their relationships to other features. Although computer technology has enhanced the behavioral aspects of these relationships, the fundamental ways that these geographic features are represented—by points, lines, and polygons—has largely remained unchanged. Esri geodatabase technology has improved the management of these points, lines, and polygons by providing tools to create geographic feature representations, enforce data integrity, and establish relationships among the geographic features that more closely model real-world situations.

As illustrated in the previous exercise, the opportunities to manage data using GIS methodology can be enhanced by careful thought and preplanning to ensure that an accurate portrayal of geographic features and their relationships is contained in the geodatabase. Preplanning the geodatabase is enhanced through a structured, organized logical data model to ensure that every conceivable relationship is accounted for in the model. This preplanning phase is no easy task. However, it is much easier to spend time at the outset designing your geodatabase than it is to change it once you've begun entering data into the model.

Organizing your geodatabase using feature classes and feature datasets allows you to refine relationships and behaviors for the data. Feature classes, as the most basic representation of geographic data in the geodatabase, can be logically grouped together to form feature datasets. Although there are many different techniques for organizing geographic data in the geodatabase, the organization of the data must be guided by the behavior of

these features in the real world. For example, if feature classes contained in the geodatabase work together to form a geometric network, represent a terrain, or establish a topology, the feature classes must reside in the same feature dataset. Such behaviors among the data must be considered while designing the geodatabase.

Once your design is complete, using domains for your attribute data and other techniques will reduce costly mistakes during the data entry phase of your geodatabase's development. Additional techniques provided by the geodatabase, such as the creation of subtypes, optimize how data is organized and utilized within the geodatabase. Using the many tools available within your project, and with a thoroughly planned approach, your new geodatabase will adequately portray the geographic features and associated relationships among them. As a result, your model of reality as contained in the geodatabase will represent the real-world features as closely as possible.

STUDY QUESTIONS

Answers to the study questions in this book are available on the instructor resources DVD.

1. What is a logical model of a geodatabase, and why should you develop a logical data model when designing your geodatabase?
2. What are the principal advantages of using subtypes? Give one example of a situation in which you would create a subtype, and specify why.
3. What are the principal advantages of using domains?
4. What is the difference between feature classes and feature datasets? When must you create a feature dataset?

Other study topics

Search for these key phrases in ArcGIS Pro Help for further reading:

1. Fundamentals of the geodatabase
2. What is a geodatabase?
3. Introduction to attribute domains
4. Fields, domains, and subtypes

Tutorial 1-2: Creating a geodatabase–expanding the logical model

The components of a geodatabase can have various spatial relationships, or behaviors, that form a topology. These behaviors can exist among points, lines, and polygons and will impact the logical model of a database. The most efficient designs will consider topology from the beginning.

LEARNING OBJECTIVES

- Design linear feature classes
- Investigate data behavior
- Design for topology
- Design point feature classes

Introduction

The first tutorial used the geodatabase design forms to construct a logical design for a parcels database. That dataset consisted of a polygon feature class along with a linear feature class to aid in symbolizing the parcel boundaries.

In this tutorial, you will design another set of feature classes to store data for a sewer system. The process will include investigating the behavior of the data, and then trying to accommodate it in the design.

Remember to look at how the data will interact with feature classes, as well as any possible domains or subtypes that may be used. This investigation will help to build not only an efficient design but also a good model of reality.

Designing the data structure

Scenario

After your successes with the parcels and zoning datasets, Oleander's Public Works Department is seeking your help to create a geodatabase for the sewer system. You will need to design this geodatabase for them.

Sewer systems are a simple design. They consist of pipes to carry wastewater to the treatment plant. In real life, it's important that the pipes connect to ensure a direct flow route from the beginning of the system to the end. In the data model, you will also want to ensure connectivity, which will allow the data to be used later to construct a network dataset.

A great amount of data can also be stored as attributes of the linear features. Some of the basic information includes the size of the pipe, the material it's made of, and the year of installation.

This portion of the process is intended to inspire thought and creativity. If done correctly, your designs will be viable for years to come. Once all the designs are completed, they will be used to create the data structure in ArcGIS Pro.

Data

Since you are creating this geodatabase from scratch, there is no data to start with. But you will need to print the geodatabase design forms from the downloaded materials as an aid in the design process. Print as many of the pages as necessary to contain all your designs.

Tools used

- Geodatabase design forms

Begin the geodatabase design process

Using the geodatabase design forms, you will once again commit your thoughts to paper and examine all aspects of how the data will be used, edited, and symbolized. The first part of the design will be to name the geodatabase. Since the data likely will be used in a network later, it will also require a feature dataset.

1. **On the first page of a new set of design forms, write the name of the new geodatabase as** Utility_Data**. On the next line, add the name of the feature dataset as** Wastewater**.**

   The sewer lines will be built as linear features, which will require a feature class. Several attributes can be stored with the lines, as mentioned earlier. You'll add each of these attributes to the design forms.

2. **On the geodatabase design form, write the name of the new feature class,** SewerLines**, with a feature type of** LINE **and an alias of** Sewer Lines**.**

Geodatabase design forms			***ArcGIS Pro***
Geodatabase name			**Utility_Data**
Feature dataset name			**Wastewater**
Feature classes:			
	Type	Feature class name	Alias
	LINE	**SewerLines**	**Sewer Lines**

   Next, you will need to fill in the Tables worksheet and show which fields the feature class will contain. The three fields that were required by Public Works were pipe size, which can be a number; pipe material, which can be text; and the year the pipe was installed, which is also a number.

3. **On the Tables worksheet, write the name of the feature class. Then write the fields** Pipe_Size **with a data type of** SI**,** Material **with a data type of** Text**, and** Year_Built **with a data type of** LI**. Add the aliases of** Pipe Size**,** Pipe Material**, and** Year Built**, respectively.**

   In Oleander, some of the sewer lines that run through the city belong to other agencies. Some are the pipes of other cities and are headed for the treatment plant, and some belong to the regional utility that handles all the wastewater treatment for local cities. They all must be included in the dataset and on the maps to prevent accidentally digging into them.

The owner of the line must also be recorded, so you'll add a field named Description to store the name of the owner of each pipe.

4. **In the Field Name column, add a field named** Description**. Write a field type of** Text **with an alias of** Owner**.**

Tables worksheet							
Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
SewerLines	Pipe_Size	Pipe Size	SI				
	Material	Pipe Material	Text				
	Year_Built	Year Built	LI				
	Description	Owner	Text				

Data integrity issues

For this data, it is important that every pipe have an entry for size and material. However, year of construction may not be known for some of the older, existing pipes. So, do not accept null values for the fields Pipe_Size and Material, but allow nulls for Year_Built. Also, the ownership of every pipe must be known, so don't allow for nulls there.

1. **For each field, write *N* next to the aliases Pipe Size, Pipe Material, and Owner in the Nulls column. Write *Y* next to Year Built in the same column.**

The next data integrity issue is to investigate the use of domains. Sewer pipes vary in size from 6 inches to 12 inches in 2-inch increments. Pipes larger than 12 inches are called *interceptors* and are metered to determine the charge to the city. In Oleander, the interceptors are owned by a regional utility that handles all the wastewater processing. Although the pipes run through the city, Oleander's Public Works Department does no service or maintenance on them.

If a domain was built for pipe size, it could prevent some data entry errors. The choices would be to use coded values and enter a discrete set of values or use a range and give a low and a high value, such as 6 and 12. A range would allow any numeric entry between these values, and because the sizes increase in 2-inch increments, there would be values allowed by the domain that are not allowed in reality. For example, the range from 6 to 12 would allow an entry of 9, but there is no such thing as a 9-inch sewer pipe. So, using a range wouldn't work. It is apparent that a discrete list of coded values should be entered.

2. **On the Tables worksheet, write the name of the domain for the field Pipe_Size as** Sewer_Pipe_Size **with the** (D1) **notation. Then on the Domains worksheet, write the same name. Add a description of** Sewer Pipe Size**, set the field type as** SI**, and write the domain type as** Coded Values**. Enter the values as shown and their corresponding descriptions:**
   - 6 = 6"
   - 8 = 8"

- 10 = 10"
- 12 = 12"

Domains worksheet						
	Domain name	Description	Field type	Domain type	Coded values / Range Code (Min)	Value (Max)
D1	Sewer_Pipe_Size	Sewer Pipe Size	SI	Coded Values	6	6"
					8	8"
					10	10"
					12	12"

Notice that although the field stores integers, and the code must be an integer, the associated description can be text. The description will be useful in labeling the text later, as the inch marks will be visible on the labels that ArcGIS Pro generates.

Another data integrity tool is to include subtypes. Subtypes can be used to segregate data, so that there will be separate domains and defaults for each subset of data. In this scenario, the data might be separated by material. Almost all the new PVC pipes going in are 8 inches, almost all the new high-density polyethylene pipes are 10 inches, and almost all the ductile iron pipes going in are 12 inches. These three are the only materials allowed for new pipes, so if each of these materials was set up as a subtype, additional control could be added to automatically populate some of the more common fields.

One problem with this approach would be the interceptors. These pipes are typically larger than 12 inches, but the size and material change for each situation. Default values wouldn't be appropriate here, so the interceptors don't play by the same rules as the Oleander pipes. Perhaps the solution is to put them in their own feature class. They could still be edited simultaneously with the Oleander data, and they could still participate in any networks that are built, as long as they reside in the same feature dataset. Also, the fields for interceptors would be identical to the Oleander lines. You'll update the worksheets to include an additional feature class for interceptors.

3. **On the geodatabase worksheet, add the name of the new linear feature class as** Interceptors**, and give it an alias of** Interceptors**. Be sure to fill in its type as** LINE**.**

4. **Write the name of the feature class on the Tables worksheet, and duplicate all the fields from the SewerLines feature class.**

Because the interceptors don't have any regular size or material, there will be no domains or default values for these lines. With the problem solved, you can proceed to design the subtypes. A good field that could use a subtype is Material. By selecting the material, the default values will automatically populate the other fields. And if a pipe size other than the standard is used, the pipe size domain will prevent any incorrect values from being entered.

The subtype field must always be an integer, and the material field is set as text. This entry can be changed easily with an eraser.

5. **On the Tables worksheet, erase the field type for Material and enter** SI. **Also, add the name** Sewer_Line_Material **in the Subtype column on the right with an** (S1) **notation.**

Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
Material	Pipe Material	SI	N	(S1) Sewer_Line_Material		

Next, you can fill in the Subtypes worksheet for the first material, polyvinyl chloride (PVC). The default value for PipeSize will be 8 inches, and the domain designed previously should be applied to this field. The default for the description field will be Oleander. And to save a little typing, make the default for Year_Built 2010. You can change it once a year to keep up with new construction.

6. **On the Subtypes worksheet, write the name of the Subtype field,** Sewer_Line_Material. **Write the code as** 1 **and the description as** PVC. **In the Field column, write the name** Pipe_Size**, note its domain as** Sewer_Pipe_Size**, and its default value as** 8.

Note that the default value does not have the inch marks next to it. Remember that its field type is short integer, so the value entered in the database must be short integer. But the inch mark is stored in the domain description, which can be used for labeling later if necessary. The benefit here is that you can do math processes on the pipe size value, such as a selection PipeSize ≤ 8, but still have the value with an inch mark for labeling.

7. **Continuing on the Subtypes worksheet, write the names of the other fields and their default values. For the Description field, write a default value of** Oleander. **For the Year_Built field, write a default value of** 2019.

This work completes the design for the first choice of subtype. The next choice will be for the material type of high-density polyethylene (HDPE). The default size will be 10 inches, and the defaults for description and year built will be the same as before.

8. **On the next blank line of the Subtypes worksheet, write the code of** 2 **with a description of** HDPE. **In the Field column, write the name** Pipe_Size**, note its domain as** Sewer_Pipe_Size**, and its default value as** 10. **As before, write a default value of** Oleander **for the Description field and** 2019 **as the default value for the Year_Built field.**

Subtypes worksheet						
				PRESET DEFAULTS		
	Subtype name	Code	Description	Field	Domain name	Default value
S1	Sewer_Line_Material	1	PVC	Pipe_Size	Sewer_Pipe_Size	8
				Description		Oleander
				Year_Built		2019
		2	HDPE	Pipe_Size	Sewer_Pipe_Size	10
				Description		Oleander
				Year_Built		2019

YOUR TURN

Fill in the information for a third subtype code with the material type ductile iron**, or** DI**. It will have a default size of** 12 inches **with the same domain as the other sizes, as well as a default description of** Oleander **and default year built of** 2019**.**

There are two more material types, and although they are no longer installed new, they could cause validation problems later if they are not included in the design. The two types are concrete and clay. You'll add them as choices 4 and 5 on the Subtypes worksheet. They will require no domains or defaults because they will not be used to enter new pipes.

9. **On the next blank line, write a code** 4 **with a description of** Conc **and a code** 5 **with a description of** Clay**. No defaults or domains are required for these subtypes.**

Subtypes worksheet						
				PRESET DEFAULTS		
	Subtype name	Code	Description	Field	Domain name	Default value
S1	Sewer_Line_Material	1	PVC	Pipe_Size	Sewer_Pipe_Size	8
				Description		Oleander
				Year_Built		2019
		2	HDPE	Pipe_Size	Sewer_Pipe_Size	10
				Description		Oleander
				Year_Built		2019
		3	DI	Pipe_Size	Sewer_Pipe_Size	12
				Description		Oleander
				Year_Built		2019
		4	Conc			
		5	Clay			

This work completes the design for linear features. Next will be investigating the point features associated with the sewer lines. At each intersection of sewer lines, and at various locations along their length, manholes are constructed for maintenance. At the ends of the lines, a smaller access port called a *cleanout* is added to accommodate the mechanical device that is run down the pipes to clean out clogs. The cleanouts will be represented in the geodatabase by points, with certain attributes associated with them.

These points have a behavior relationship with the lines, in that they must fall on top of the lines. If any networking is done, the points must be snapped to the lines to preserve connectivity. They must also reside in the same feature dataset.

The data associated with the points will include several fields. The first will be a code that marks which points are manholes and which are cleanouts. Other information such as the flow line, rim elevation, and depth (rim elevation minus the flow line elevation) will be copied from the construction documents. Finally, a field for the year of construction and another for description (ownership) will be needed.

10. **On the geodatabase design form, write the name of the new point feature class as** SewerFixtures**. Give it a feature class type of** PNT **(Point) and a description of** Sewer Fixtures**.**

    Next, add the fields for the point feature class on the Tables worksheet.

11. **On the Tables worksheet, write the name of the new feature class, and add the following field names with their field types, aliases, and null value allowances:**
    - Fix_Type, Fixture Type, SI, No
    - Flowline, Flowline Elevation, Float, Yes
    - Rim_Elev, Rim Elevation, Float, Yes
    - Depth, Depth from Surface, Float, Yes
    - Year_Built, Year Built, LI, Yes
    - Description, Owner, Text, No

    Even though there won't be defaults or domains for any of these fields, it might be useful to make fixture type a subtype. One benefit of subtypes is that each code in the subtype list can be selected as a target when editing. Without a subtype, you would set the target as SewerFixtures and click to add a point. Then you would have to set the fixture type immediately because it cannot be null. With a subtype set for fixture, the target drop-down list would show the two types of fixtures allowed. As each new point is entered, the fixture type field is automatically populated, meaning that it can never be null. For the short amount of time it takes to set up the subtype structure, it would be a great way to enforce the data integrity rule of not allowing null values. At the same time, a default for description and year built could be added for convenience. You'll start by noting the subtype name on the tables form, and then populate the subtype form.

12. **On the Tables worksheet, write the subtype name** Sewer_Fix_Type **for the Fix_Type field with a notation of** (S2)**. Under SewerFixtures, for the following fields, add a default Year Built value of** 2019 **and a default Description value of** Oleander**.**

Tables worksheet							
Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
SewerFixtures	Fix_Type	Fixture Type	SI	N	(S2) Sewer_Fix_Type		
	Flowline	Flowline Elevation	Float	Y			
	Rim_Elev	Rim Elevation	Float	Y			
	Depth	Depth from Surface	Float	Y			
	Year_Built	Year Built	LI	Y		2019	
	Description	Owner	Text	N		Oleander	

13. **On the Subtypes worksheet, write the name of the subtype as** Sewer_Fix_Type**. Give it a code** 1 **for** Manhole **and a code** 2 **for** Cleanout**.**

The interceptors will also have associated fixtures, but they are all manholes. This factor makes for a simple feature class because all the features will be symbolized the same. They will have the same fields as the Oleander fixtures, except for fixture type and description. These fields are unnecessary because their values would always be the same.

14. **Add a new feature class to the Geodatabase worksheet (page 1). Name it** InterceptorFix**, with a feature type of** PNT **and a description of** Interceptor Fixtures**.**

15. **Fill in the Tables worksheet for the new feature class** InterceptorFix **with these fields and values:**
    - Flowline, Flowline Elevation, Float, Yes
    - Rim_Elev, Rim Elevation, Float, Yes
    - Depth, Depth from Surface, Float, Yes
    - Year_Built, Year Built, LI, Yes **(default value of** 2019**)**

Feature class or table name	Field name	Alias	Field type	Nulls (Y/N)	Domain name or subtype field (S) or (D)	Default value	Length
InterceptorFix	Flowline	Flowline Elevation	Float	Y			
	Rim_Elev	Rim Elevation	Float	Y			
	Depth	Depth from Surface	Float	Y			
	Year_Built	Year Built	LI	Y		2019	

The geodatabase design forms make this design seem simple, but it is a fairly complex database. A good deal of thought was put into the fields required for the feature classes, the relationships of the feature classes, and the inclusion of data integrity rules, such as defaults, domains, and subtypes.

Review the design forms and resolve any questions that you may have, because the next tutorial, 2-1, will have you build these data structures in ArcGIS Pro.

Exercise 1-2

The tutorial showed how to apply design strategies and data integrity rules to point and linear feature classes. Each feature type was analyzed against the reality it is supposed to model to build as much behavior and data integrity as possible.

In this exercise, you will repeat the process with storm drain data. Oleander's Public Works Department would like a geodatabase design for the storm drain system, just like the one you did for the sewer collection system. It will consist of the pipelines and fixtures associated with them. The lines are all made from reinforced concrete pipe (RCCP) and vary in size from 15 inches to 45 inches in 3-inch increments. The pipes are usually classified as laterals (21 inches or less), mains (longer than 21 inches), and boxes (square pipe with no

restriction in size). The data for these features includes a pipe size, material, description, flowline in, flowline out, slope, year installed, and a designation for a public or private line.

Connected to these pipes are various types of fixtures listed as follows. These fixtures will be used as the subtypes, and their code from the existing data is included:

101 = curb inlet
102 = grate inlet
104 = junction box
106 = Y inlet
107 = junction box/manhole
108 = outfall
109 = headwall
110 = beehive inlet
111 = manhole

The type of data collected for these features includes a description, flowline elevation, inlet size, top elevation, year built, designation for a public or private line, and a rotation angle.

Analyze the descriptions of this data and determine what feature datasets and feature classes need to be created, what fields they should contain, any domains that might need to be created, and any subtypes that might be beneficial.

- Print a set of the geodatabase design forms as necessary.
- Use the forms to create the logical model of feature classes for the zoning polygons and zoning boundaries.
- Investigate the use of domains and subtypes to build data integrity and behavior into your design.

Getting started

Here's a little help to get started:

- Decide how many feature classes you want to create.
- List the fields that will need to be in each feature class.
- Determine the field type, null status, and default value for each field.
- Investigate the use of domains for these fields.
- Look for fields that describe a "type" or "category" that could be used as a subtype, such as fixture type.

WHAT TO TURN IN

If you are working in a classroom setting with an instructor, you may be required to submit the design forms you created in tutorial 1-2.

- The completed design worksheets for:
  - Tutorial 1-2
  - Exercise 1-2

Review

Whereas the first tutorial focused on the development of a geodatabase to contain parcel-related information, this tutorial focused on the development of a geodatabase to represent a sewer system. Both tutorials focus on real-world examples of the types of critical information managed by every city, town, or other local governmental entity. Given the tremendous municipal resources dedicated to the management of these systems, an adequate foundation for managing them is essential. The geodatabase allows for physical information, as well as information on the behaviors among the components of these systems, to be accurately portrayed in a GIS. By using the combination of good information and good data integrity controls (behavior), the geodatabase enhances effective and efficient decision-making capabilities.

One of the most critical steps in developing a comprehensive geodatabase is the initial planning phase. Although we have focused on and discussed GIS functionality within both tutorials, we have yet to even start a create process in the software! Planning for a geodatabase development project involves both gathering the physical system requirements and understanding how interrelated objects behave in the physical system. To further enhance your design, you will also need to know what kinds of questions your customers will need to have answered. Once an adequate knowledge of the system is gained, it is then up to the skill of the geodatabase developer to build these capabilities into the corresponding geodatabase.

A properly designed geodatabase will be worthless to the engineers and others who rely on it to represent the real world if it is full of errors. The geodatabase allows the designer to apply *domains* to feature attributes to ensure that the correct information is correctly recorded within the geodatabase. Flexibility to adequately portray and control features and their behavioral characteristics within the model is afforded using *subtypes* of features. Interrelated behavior among features is enhanced using *topology*. The success of a geodatabase often depends on a thorough knowledge of how and when to apply these data integrity tools.

STUDY QUESTIONS

1. What is topology, and why is it an important concept when designing a logical model of a geodatabase?
2. Think about the principal ways in which features are represented spatially within GIS. Give an example of each feature type, a possible domain for each type, and possible subtypes for each type. Explain your rationale.
3. Why is it important to fully understand a system to be represented and managed in GIS? How do you determine its structure?

Other study topics

Search for these key phrases in ArcGIS Pro Help for further reading:

1. Introduction to subtypes
2. Geoprocessing considerations for attribute domains
3. Geoprocessing considerations for subtypes