Understanding Cursors in ArcObjects

Learn how to use cursors to manipulate records in feature classes and tables.

By Eric Pimpler, President, GeoSpatial Training & Consulting, LLC

When you hear the term *cursor,* what comes to mind? Is it the symbol on a screen that shows where the next input activity will take place? In ArcObjects, a cursor refers to a subset of records that is obtained by applying an attribute and/or spatial query on a feature class or table. This subset of records is held in memory rather than visually displayed. Do not confuse cursors with selection sets. Selection objects are used to display the currently selected features or rows in the ArcMap display, while cursors are not used for display purposes.

For instance, a Search Cursor could be used to programmatically generate a mailing list of all parcels of land within a 100-year floodplain that have a property value greater than $100,000. ArcObjects provides the ability to obtain cursors from geographic datasets (FeatureClasses) as well as regular database tables. These cursor objects allow you to manage a subset of records in a single object. This article will explore the ArcObjects classes, methods, and properties used to manipulate cursors.

Cursors versus FeatureCursors

ArcObjects uses distinct classes to manage subsets of records depending on the data source. Cursors and FeatureCursors are very similar objects except that Cursors operate on Table objects and FeatureCursors operate on FeatureClasses. In other words, Cursors are class structures built for the specific purpose of working with subsets of records stored in traditional database tables, while FeatureCursors are built specifically for working with subsets of records stored in geographic data structures such as shapefiles, personal geodatabases, and enterprise geodatabases.

Types of Cursors

There are three types of cursors found in both the Cursor and FeatureCursor classes. The most commonly used type of cursor is the Search Cursor. It is used in query operations to return a subset of records that meet the query conditions. Search Cursors are read-only cursors that you can iterate through to obtain information.

You cannot use a Search Cursor to insert, update, or delete records from a table. A second type of cursor, the Insert Cursor, is specifically used to insert a new record in a table. The Update Cursor is used to update or delete records in a table. The records returned in an Update or Search Cursor can be constrained to match attribute criteria and/or spatial criteria.

It is important that you create the proper type of cursor for the operation that you are performing. For example, don't create a Search Cursor if you are attempting to update data in a table. As previously mentioned, Search Cursors are read-only structures so you won't be able to update the data. Each cursor type will be explored in more detail later in this article.

Cursor Class

As mentioned previously, the Cursor class is used to create objects that work with database tables. The Cursor class in ArcObjects is an instantiable class. This means you must use another object to obtain an instance of this class. In this case, the Table class in ArcObjects is used to create an instance of the Cursor class. The Table class contains three methods that can be used to return an instance of the Cursor class. The type of cursor returned is dependent upon the method called. Figure 1 shows the

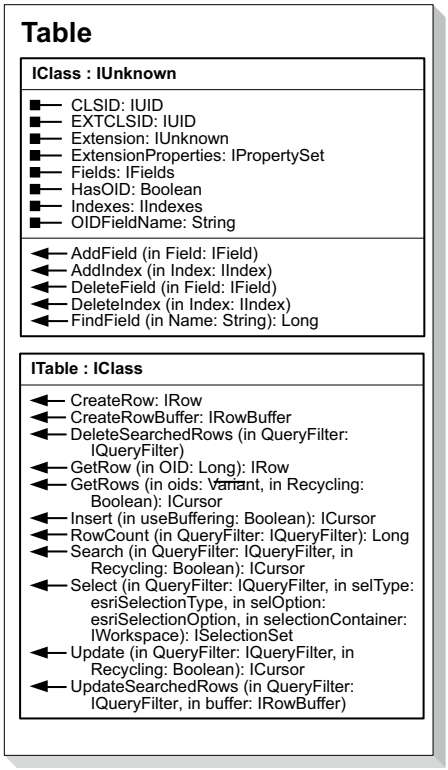

Figure 1: Object model diagram for the Table class

object model diagram for the Table class in ArcObjects. The ITable interface has three methods that can be used to return specific types of cursor objects. The Search, Insert, and Update methods on ITable are used to return cursor instances. The names of the methods correspond to the type of cursor returned.

After one of these methods has been called, ArcObjects returns an instance of ICursor. Figure 2 shows the object model diagram for the Cursor class. Search, Insert, and Update all return an instance of ICursor. ICursor has one property (Fields) and a number of methods that can be used to manipulate the subset of records. Some of the methods available on ICursor may not be applicable depending on the type of cursor that you are working with. For instance, if you created a Search Cursor, the InsertRow and UpdateRow methods will return an error if called since you are not working with Insert or Update Cursors.

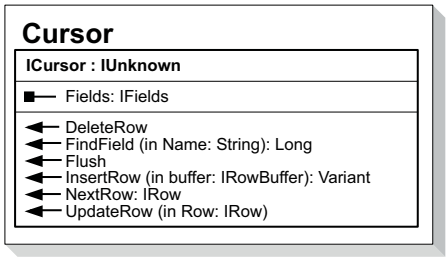

Figure 2: Object model diagram for the Cursor class

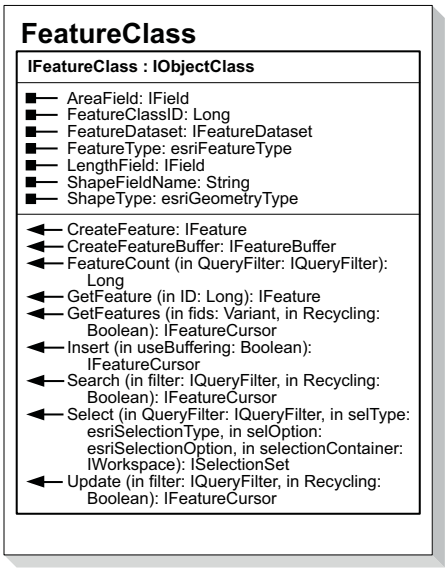

Figure 3: Object model diagram for the FeatureClass

FeatureCursor Class

The FeatureCursor class is very similar to the Cursor class with the exception that FeatureCursors are used when you're working with geographic datasets rather than traditional database tables. Geographic datasets are typically shapefiles and geodatabases in the form of an ArcObjects FeatureClass. Similar to the Cursor class, the FeatureCursor class is an instantiable class created through the use of a method on a FeatureClass object. Similar to the ITable interface, the IFeatureClass interface contains Search, Insert, and Update methods that can be used to return an instance of IFeatureCursor.

After one of these methods has been called, an instance of IFeatureCursor will be returned. The properties and methods available on IFeatureCursor are functionally identical to those on ICursor, although the method names differ slightly, for instance, InsertFeature versus InsertRow.

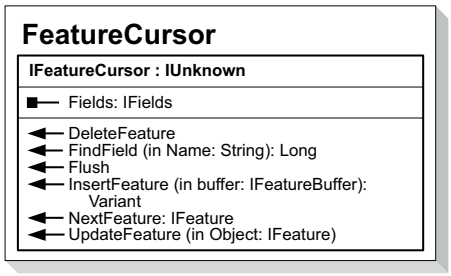

Figure 4: Object model diagram for FeatureCursor

Applying Attribute and Spatial Constraints

Look at the object model diagrams for FeatureClass and Table and look at the Search, Insert, and Update methods. Notice that each method is used to return a cursor that contains a parameter specifying an instance of IQueryFilter. Note the IQueryFilter parameter. IQueryFilter is an object that can be created to constrain the subset of records that is created in memory.

For instance, if you are querying a parcel geodatabase, you might want to constrain the results to parcels with a value greater than $100,000. You can apply this type of constraint through the IQueryFilter interface. In addition, when working with FeatureClass objects, you can also apply an optional SpatialFilter through the ISpatialFilter interface. You could return all parcels within a floodplain (using the spatial filter) that had a property value of greater than $100,000 (using an attribute query). Remember that spatial filters can only be applied to a FeatureClass. Attempting to apply a spatial filter to a database table will result in an error because there is no geographic component to which the filter can be applied. Let's look at the QueryFilter and SpatialFilter classes in more detail.

QueryFilter

Before producing a Cursor or FeatureCursor from a dataset, you can define a QueryFilter that specifies criteria limiting the number of records returned. Because QueryFilter is a creatable class, you can use the New keyword in Visual Basic for Applications to create an instance of this class. Typically you will work with the IQueryFilter interface on the QueryFilter class to define an attribute constraint. The WhereClause property is used to limit the query. The code sample in Figure 5 could be used to limit the parcels returned to only those with a value greater than $100,000.

```
Dim pQueryFilter as IQueryFilter
Set pQueryFilter = New QueryFilter
pQueryFilter.WhereClause = "Prop_Val >= 100000"
```

Figure 5: Limiting parcels returned using the WhereClause

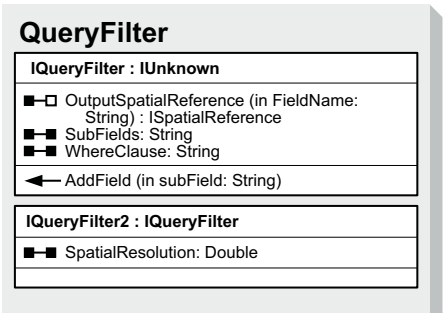

Figure 6: Object model diagram for QueryFilter

The QueryFilter (Figure 6) could then be applied to a Table or FeatureClass as seen in the code in Figure 7.

SpatialFilter

A SpatialFilter (Figure 8) can be applied to produce a subset of records based on spatial criteria. It can be applied to FeatureClasses but not Tables. SpatialFilter is a creatable class so the New keyword can be used to create an instance of this class. SpatialFilter uses a Geometry property and a SpatialRel property to define the search criteria. The Geometry property is used to specify a particular geographic feature. SpatialRel can be set to one of a predefined set of constants such as intersects, overlaps, or touches.

Because SpatialFilter is a type of QueryFilter, it also has access to all the methods and properties on that class. Therefore, you can use the WhereClause property on IQueryFilter to combine spatial and attribute constraints. See the code sample in Figure 9 for an

Continued on page 30

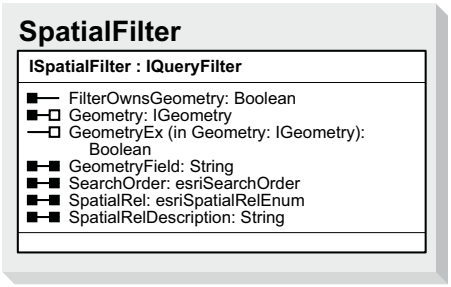

Figure 8: Object model diagram for SpatialFilter

```
Dim pCursor As IFeatureCursor
Set pCursor = pFeatureClass.Search(pQueryFilter, True)
```

Figure 7: Applying QueryFilter to a FeatureClass

Understanding Cursors in ArcObjects

Continued from page 29

example of how you can combine QueryFilter and SpatialFilter to apply spatial and attribute constraints in a single query.

Accessing Records in a Cursor

Now that you have a good understanding of the general mechanics of creating cursors, let's look at how you can access the records returned in a cursor. Remember that cursors are just an in-memory collection of records returned from a Table or FeatureClass.

When a cursor is first created, an associated pointer is also created. You access records in a cursor one row at a time. The pointer helps keep track of which row is currently being accessed. Upon initialization, the pointer actually sits above the first record. To get to the first row in the cursor, you must make a call to the NewRow (Table) or NextFeature (FeatureClass) method. These two methods advance the pointer to the next record in the cursor. The first time these methods are called, the pointer advances to the first record. Each additional call to these methods returns the next record.

At some point you will reach the end of the records available in the cursor. Any additional call to NewRow or NextFeature will return the Nothing object, indicating that the end of the cursor has been reached. Cursors in ArcObjects are forward-moving objects that will not allow you to move backward through the cursor. Once you have advanced to record two in the cursor, you can't go back to record one.

```
Dim pSpatialFilter As ISpatialFilter
Set pSpatialFilter = New SpatialFilter
Set pSpatialFilter.Geometry = pFloodPolygon
pSpatialFilter.SpatialRel = esriSpatialRelContains
pSpatialFilter.WhereClause = "prop_val > 100000"
Set pFCursor = pCustomerLayer.Search(pSpatialFilter,True)
```

Figure 9: Apply spatial and attribute constraints in a single query using both QueryFilter and SpatialFilter.

Conclusion

ArcObjects cursor structures provide you with the ability to query, insert, update, and delete records from FeatureClasses and Tables. These easy-to-create and flexible cursor structures are in-memory collections of records that can be constrained through the use of filters applied through the QueryFilter and SpatialFilter classes. Once generated, these cursor structures provide an easy-to-navigate, forward-moving structure that can be used to investigate the contents of individual records. For more information, contact

Eric Pimpler
President, GeoSpatial Training & Consulting, LLC
E-mail: eric@geospatialtraining.com
Web: www.geospatialtraining.com
Tel.: 210-260-4992