The Developers API to the Geodatabase

Sud Menon

Mike Frith

The Geodatabase API

- **Geodatabase Data Access Objects**

- **A set of COM objects and interfaces**
  - **Part of esricore.olb**
- **Exposes the complete Geodatabase data model**
- **Application programs can be written in any COM aware language :**
  - **VB, C++, ...**

Geodatabase Data Access Objects

- **Allows application programs to**
  - **Define, Maintain, Query and Edit all of the objects in the database**
- **Gives the application an object-relational view of the database.**

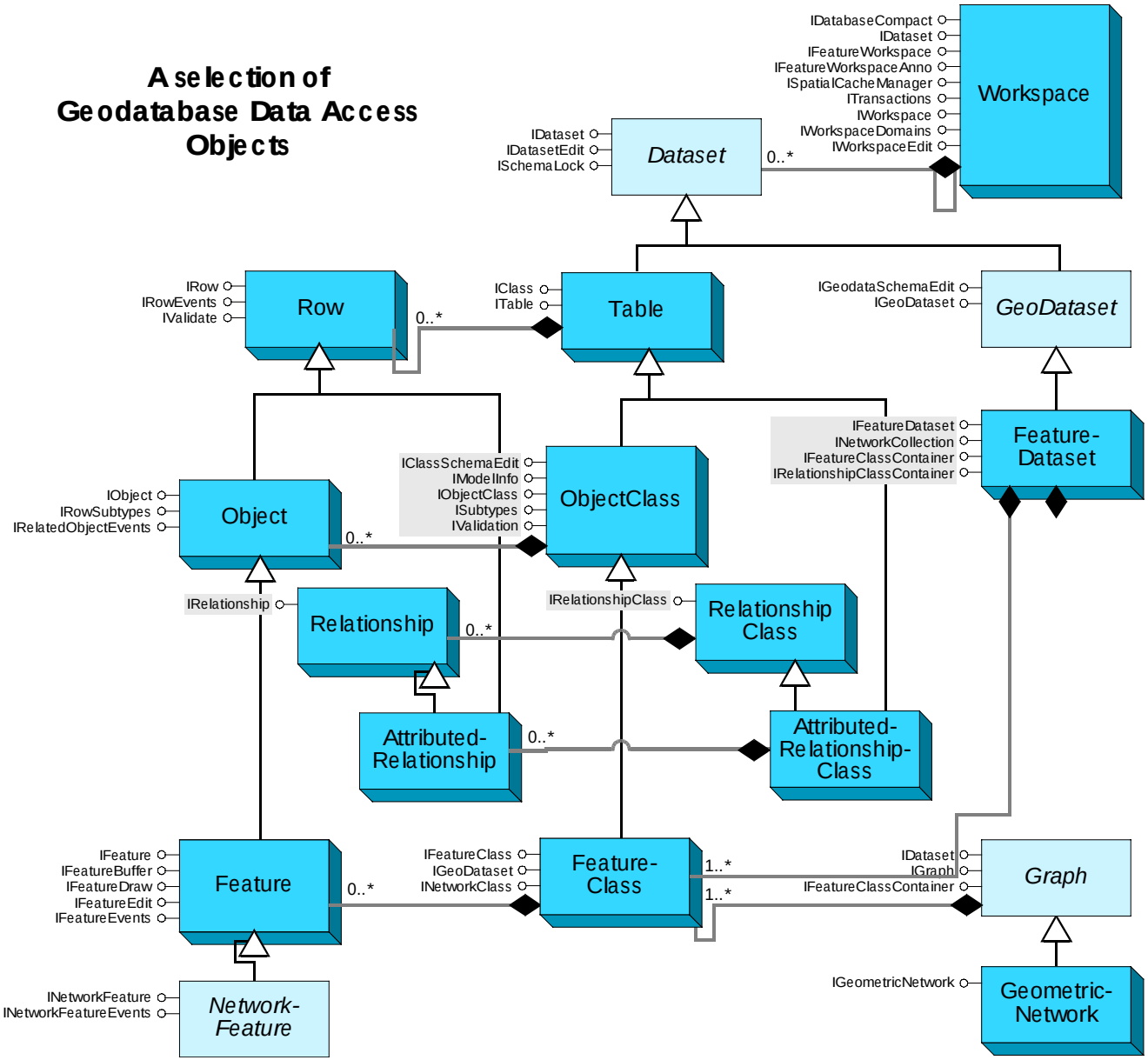
ADO based access to the Geodatabase

- Microsoft API for Universal Data Access
  - connection, command, recordset
- available using the ESRI OLEDB Provider
- Implemented on top of Geodatabase Data Access Objects
- Can access tables and feature classes using ADO or OLEDB
  - geometry return as an OGIS WKB
- See tech. session on ADO and OLEDB

Key Geodatabase Data Access Objects :

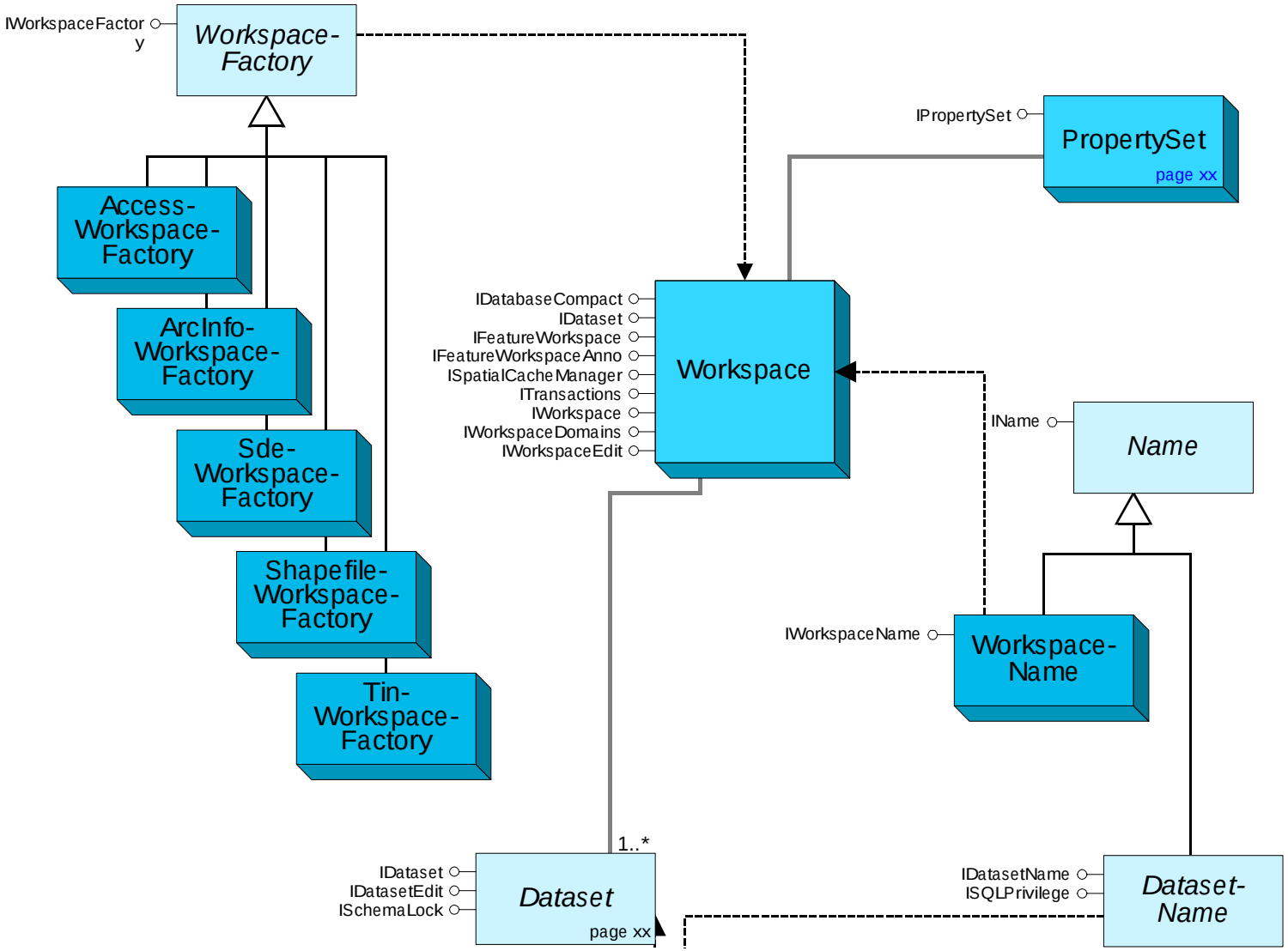
- Workspaces and Workspace Factories
- Datasets
- DatasetNames
- Tables, ObjectClasses and Feature Classes
- Rows, Objects and Features
- Cursors and Selections
- Relationship Classes and Relationships
- Edit Sessions
- Versioned Workspaces and Versions

A selection of Geodatabase Data Access Objects



Workspace Factory

- A Workspace Factory allows you to connect to a workspace
  - A Workspace represents a database
- Connection properties are specified using a Property Set object
- Connection Properties can be saved to a connection file
- Workspace Factory is a singleton, cocreatable
- A Workspace Factory keeps a cache of connected workspaces


IWorkspaceFactory
Workspace-Factory
Access-Workspace-Factory
ArcInfo-Workspace-Factory
Sde-Workspace-Factory
Shapefile-Workspace-Factory
Tin-Workspace-Factory
IPropertySet
PropertySet
page xx
IDatabaseCompact
IDataset
IFeatureWorkspace
IFeatureWorkspaceAnno
ISpatialCacheManager
ITransactions
IWorkspace
IWorkspaceDomains
IWorkspaceEdit
Workspace
IName
Name
IWorkspaceName
Workspace-Name
1..*
IDataset
IDatasetEdit
ISchemaLock
Dataset
page xx
IDatasetName
ISQLPrivilege
Dataset-Name

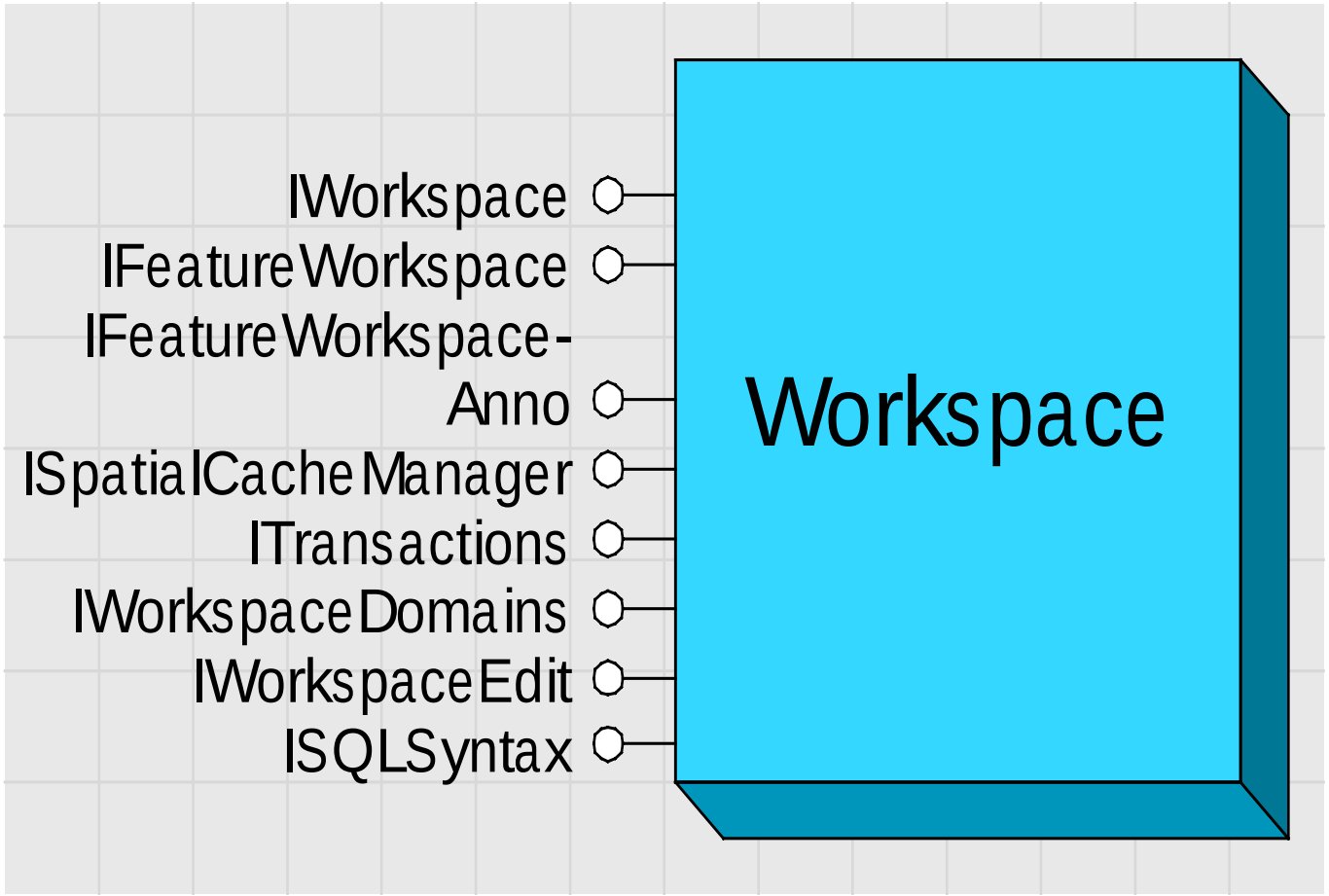
Workspace Factory

- IWorkspaceFactory
  - *Open*
    - Opens a workspace given a connection property set
  - *OpenFromFile*
    - Opens a local database or file system workspace given a file system pathname to the workspace (eg .mdb)
    - Opens a remote database workspace given a file system pathname to a connection file
  - *Create*
    - Creates a local database or file system workspace
    - Creates a connection file for a remote database workspace

Workspace

- **A Collection of Datasets**
- **A Collection of Domains**
- *Workspace Type* **:**
  - *FileSystem* **(eg. Shapefile Workspace)**
  - *LocalDatabase* **(eg. Access Geodatabase)**
  - *RemoteDatabase* **(eg. SDE Geodatabase)**

Workspace


IWorkspace
IFeatureWorkspace
IFeatureWorkspace-
Anno
ISpatialCacheManager
ITransactions
IWorkspaceDomains
IWorkspaceEdit
ISQLSyntax
Workspace

Workspace

- IWorkspace
  - Methods to browse datasets
- IWorkspaceDomains
  - Methods to access and create Domains
- IFeatureWorkspace
  - Methods to open and create Tables, FeatureClasses, RelationshipClasses, …
- IWorkspaceEdit
  - Methods to manage an edit session on a workspace

Workspace

- ISpatialCache
  - Methods to cache features within a specified extent
- ITransactions, ITransactionOptions
  - Methods to begin and end short (DBMS) transactions
- ISQLSyntax
  - Methods to parse and assemble qualified names, SQL syntax info

Dataset

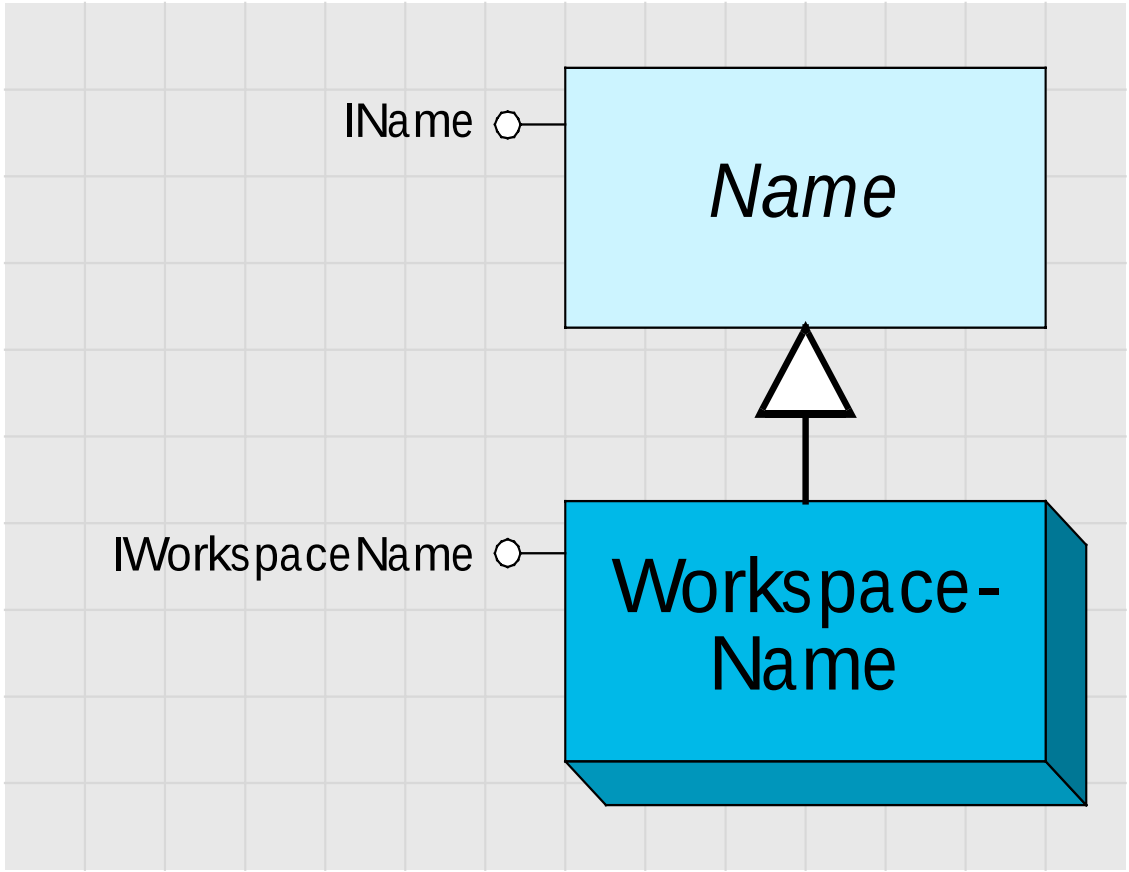
- **Abstract Class**
- **Represents a named collection of data**
- *DatasetType*
  - **esriDTTable, esriDTFeatureClass, esriDTRasterBand, …**
- *Name* **– the Qualified Name**
  - **“Tom.Parcels”**
- *FullName* **– the persistable Name Object**
- *Subsets* **– datasets contained within this dataset**
- *Workspace* **– the containing Workspace**

GeoDataset

- **Abstract Class**
- **A geographic dataset**
- **IGeoDataset :**
  - *Extent*
  - *SpatialReference*

Name

- Abstract Class
- Represents the name of an object
- Open method binds to the actual object
- Similar to a monicker
- Persistable in a binary stream


IName
Name
IWorkspaceName
Workspace-
Name

Workspace Name

- **Cocreatable**
- **IWorkspaceName :**
- **<>** *ConnectionProperties / Pathname*
- **<>** *WorkspaceFactoryProgID*
- ***Open* - returns a Workspace**
- **<** *WorkspaceFactory*
- **<** *WorkspaceType*

Dataset Name

- **Abstract**
- **IDatasetName :**
- **<>** *Name* **– the Qualified Name**
  - **“Tom.Parcels”**
- **<>** *WorkspaceName*
- **<** *Type* **– the dataset type**
- **<** *SubsetNames* **– the names of datasets contained in this dataset**

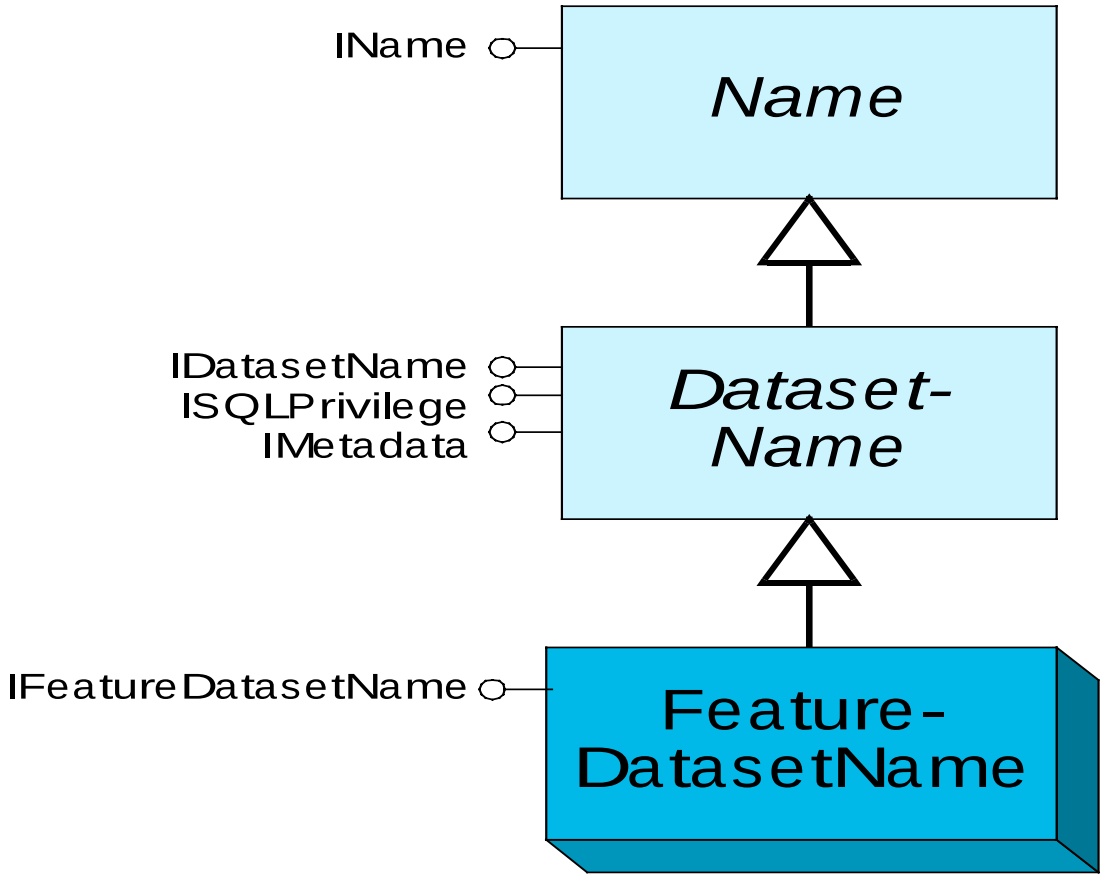
Dataset Name

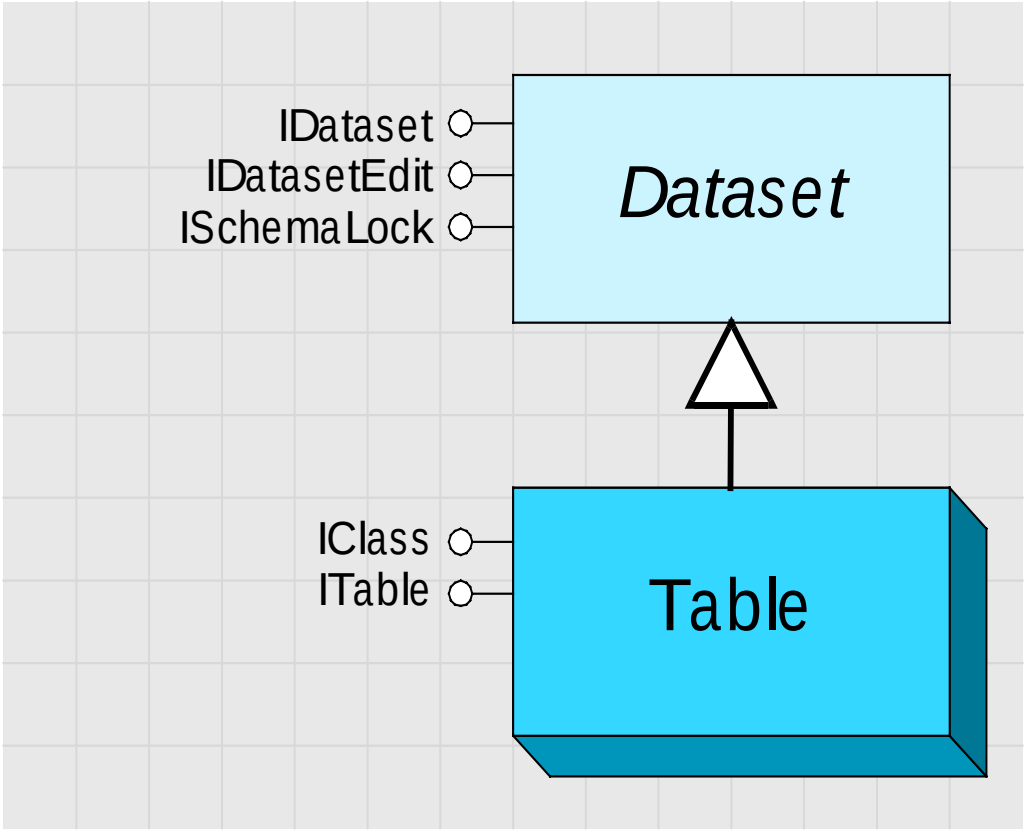
- ISQLPrivelege
- RelationshipClassName
  - Methods to grant and revoke priveleges on datasets
- IMetadata
  - Methods to get and set the metadata for a feature dataset

Dataset Names

- FeatureDatasetName
- TableName
- FeatureClassName
- GeometricNetworkName
- ....

FeatureDatasetName




IDataset
IDatasetEdit
ISchemaLock
Dataset
IClass
ITable
Table

Table

- **Abstract Class**
- **A collection of unordered rows**
- **RDBMS view**
  - **Represents an *RDBMS Table* or *RDBMS View***
- **OO view**
  - **Represents an *Object Class* or a *Relationship Class***
- **Identified by a qualified name**
- **A Table hands out row objects.**

IClass - supported by all Tables

- *CLSID* **– the COM classid for the objects handed out by the table**
  - **CLSID_Row, CLSID_Object, CLSID_Feature, CLSID_AttributedRelationship, …**

- *EXTCLSID* **- the COM classid for the class extension associated with this table**

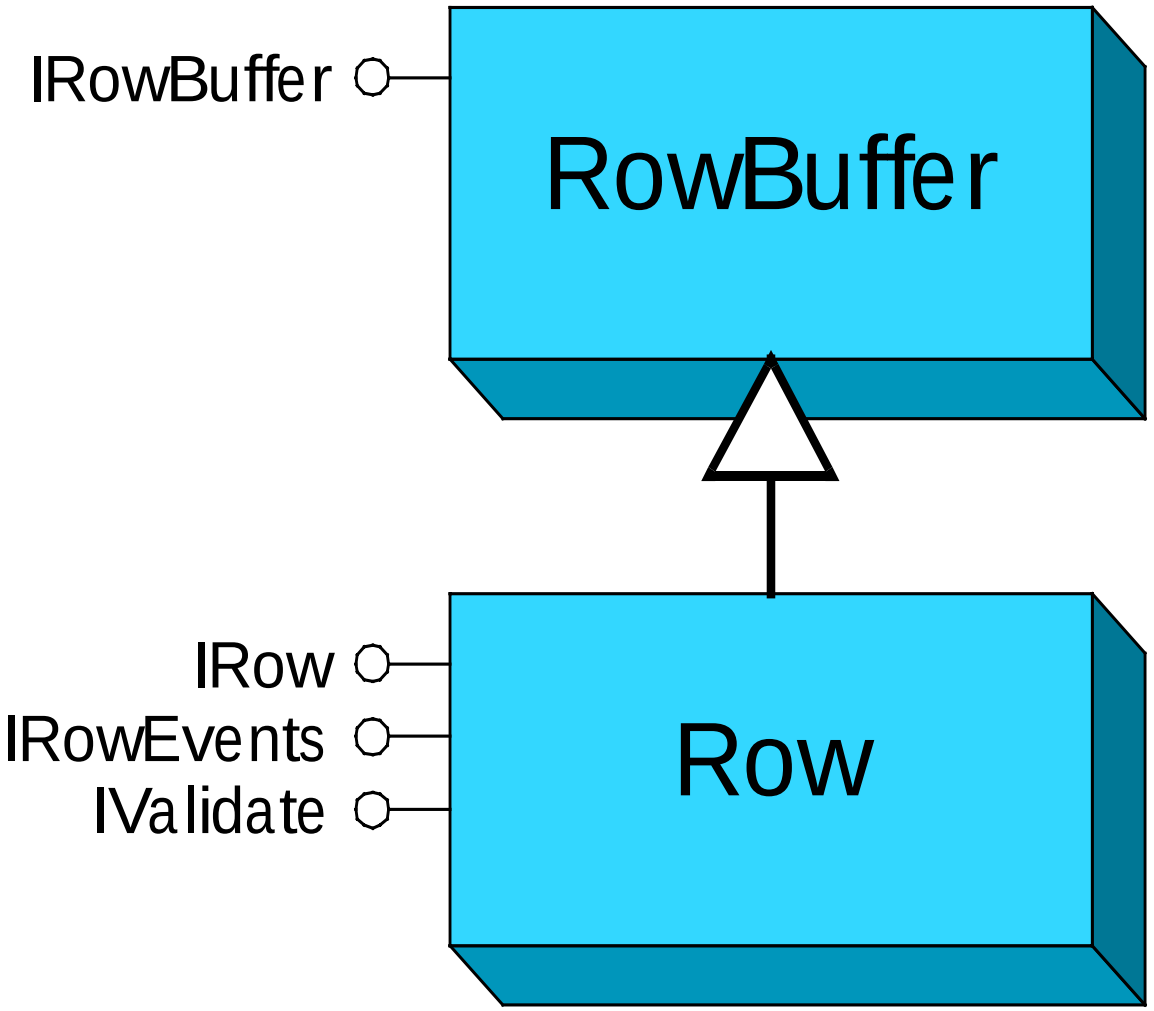
- *HasOID*, *OIDFieldName* **– object id info**
- *Fields* **– the set of fields for this table**
- *Indexes* **– the set of indexes for this table**

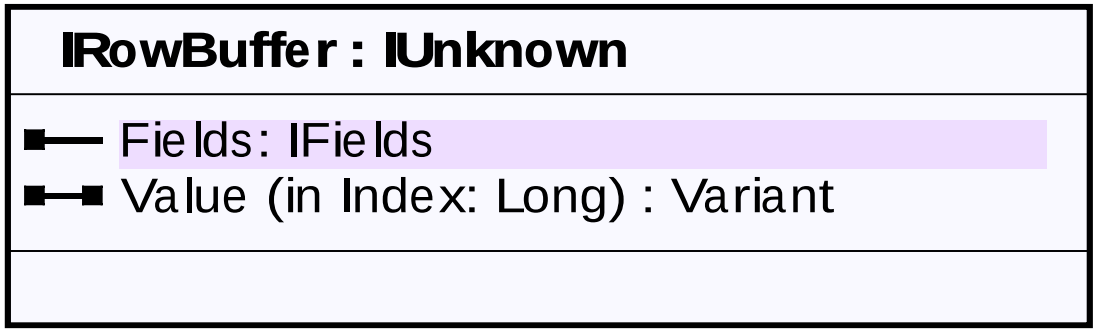
ITable

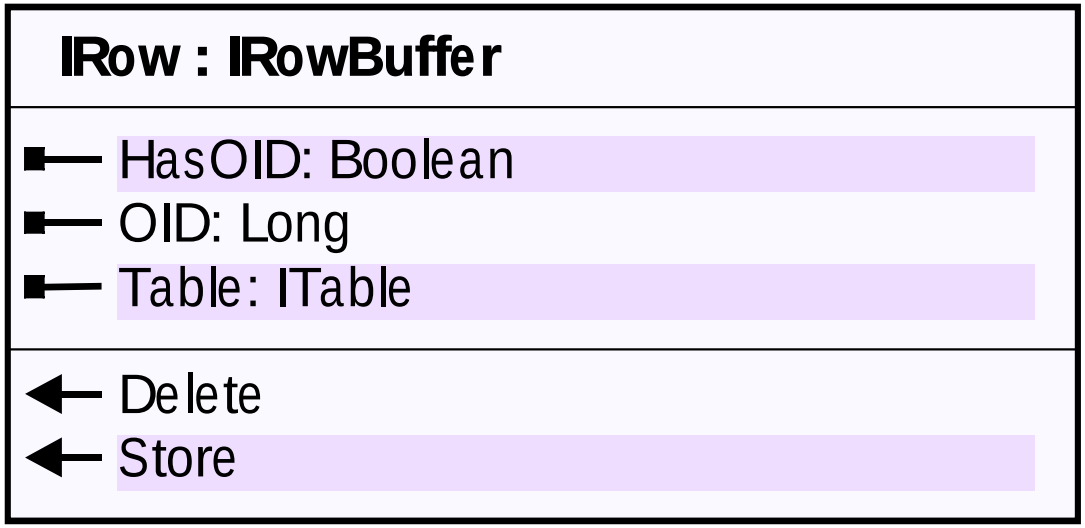
- **Supported by all tables**
- ***GetRow***
  - **Gets an existing row object given its oid value**
- ***GetRows***
  - **Returns a cursor on a set of rows given an array of oid values**

ITable

- ***CreateRow***
  - **Creates a new row in the database with system assigned object id value**
- ***CreateRowBuffer***
  - **Creates a new row buffer – no row is added to the database. The row buffer object handed back does not have an object id.**
  - **Use as argument to an insert cursor**


IRowBuffer
RowBuffer
IRow
IRowEvents
IValidate
Row

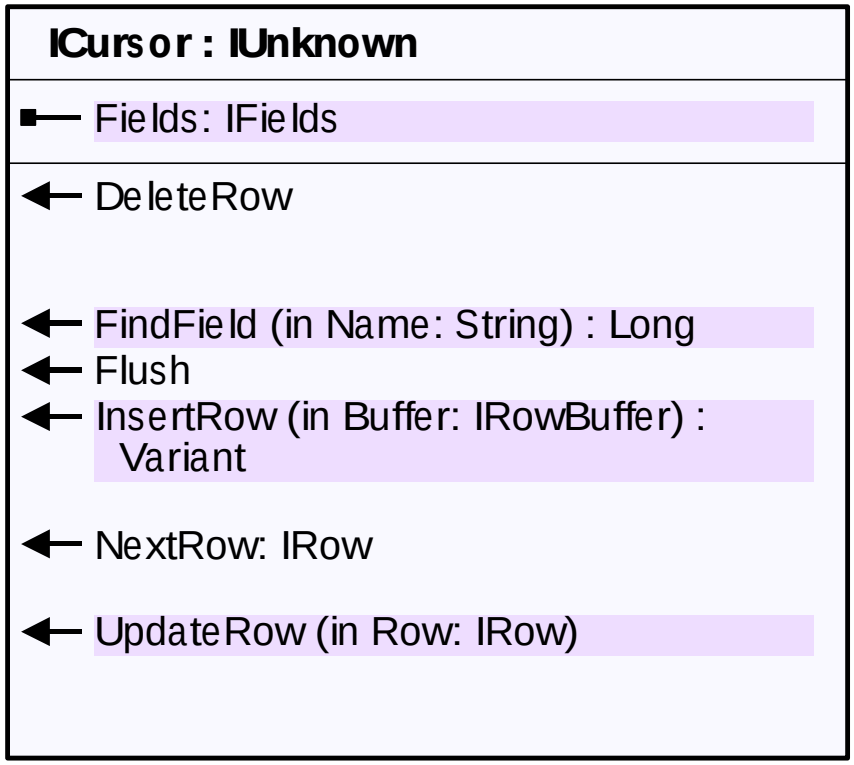

IRowBuffer : IUnknown
Fields: IFields
Value (in Index: Long) : Variant


IRow : IRowBuffer
HasOID: Boolean
OID: Long
Table: ITable
Delete
Store

ITable

- ***Search***
  - Returns a ‘search’ cursor that can be used to retrieve rows
  - Recycling cursors rehydrate a single row object on each fetch (eg. for drawing)
  - Non Recycling cursors return a separate row object on each fetch (eg. for editing)
  - Retrieved row objects may be updated and stored or deleted with polymorphic behavior
    - Eg. calling Delete on a simple row vs on a network feature

```
‘ Assume t is an ITable interface on some Table
Dim c as ICursor
Set c = t.Search Nothing, False
Dim r as IRow
Set r = c.NextRow
While not r is Nothing
        ‘do stuff with r
        ‘ r.Store
        Set r = c.NextRow
Wend
```

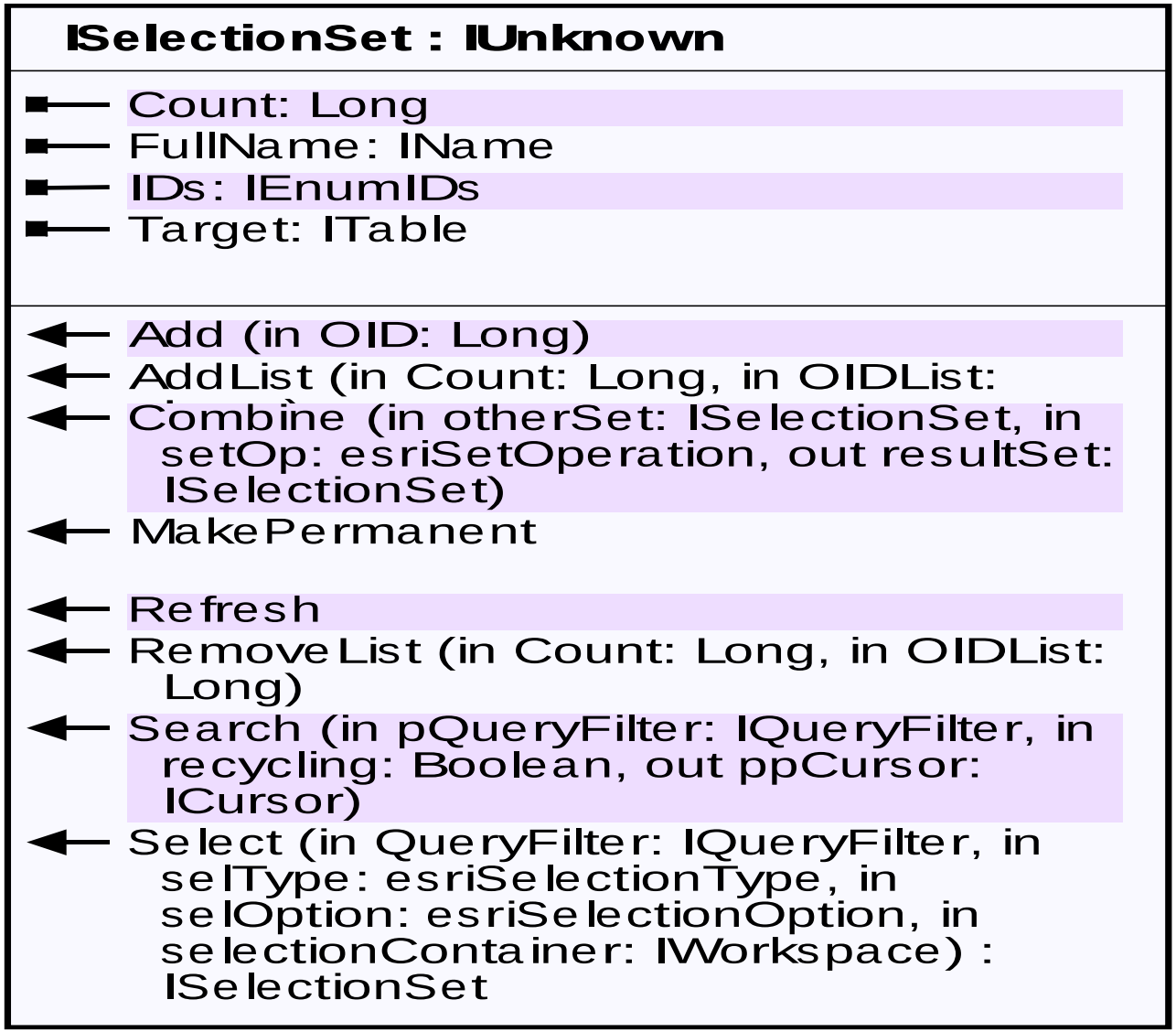

ICursor : IUnknown
Fields: IFields
DeleteRow
FindField (in Name: String) : Long
Flush
InsertRow (in Buffer: IRowBuffer) : Variant
NextRow: IRow
UpdateRow (in Row: IRow)

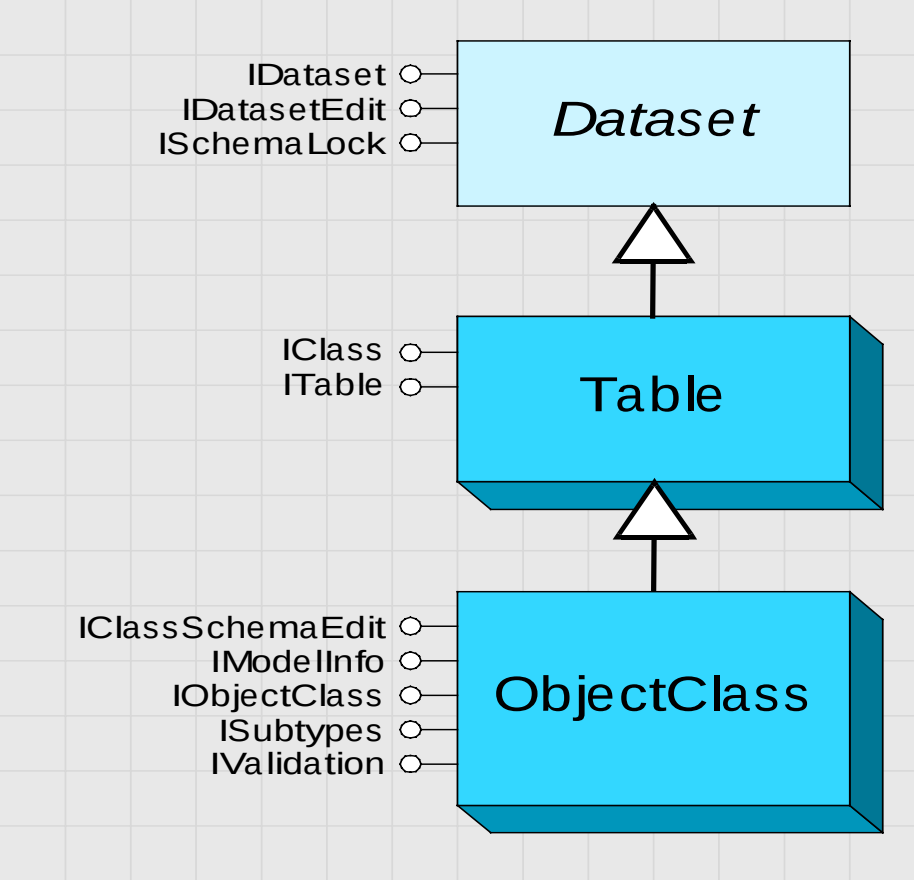
ITable

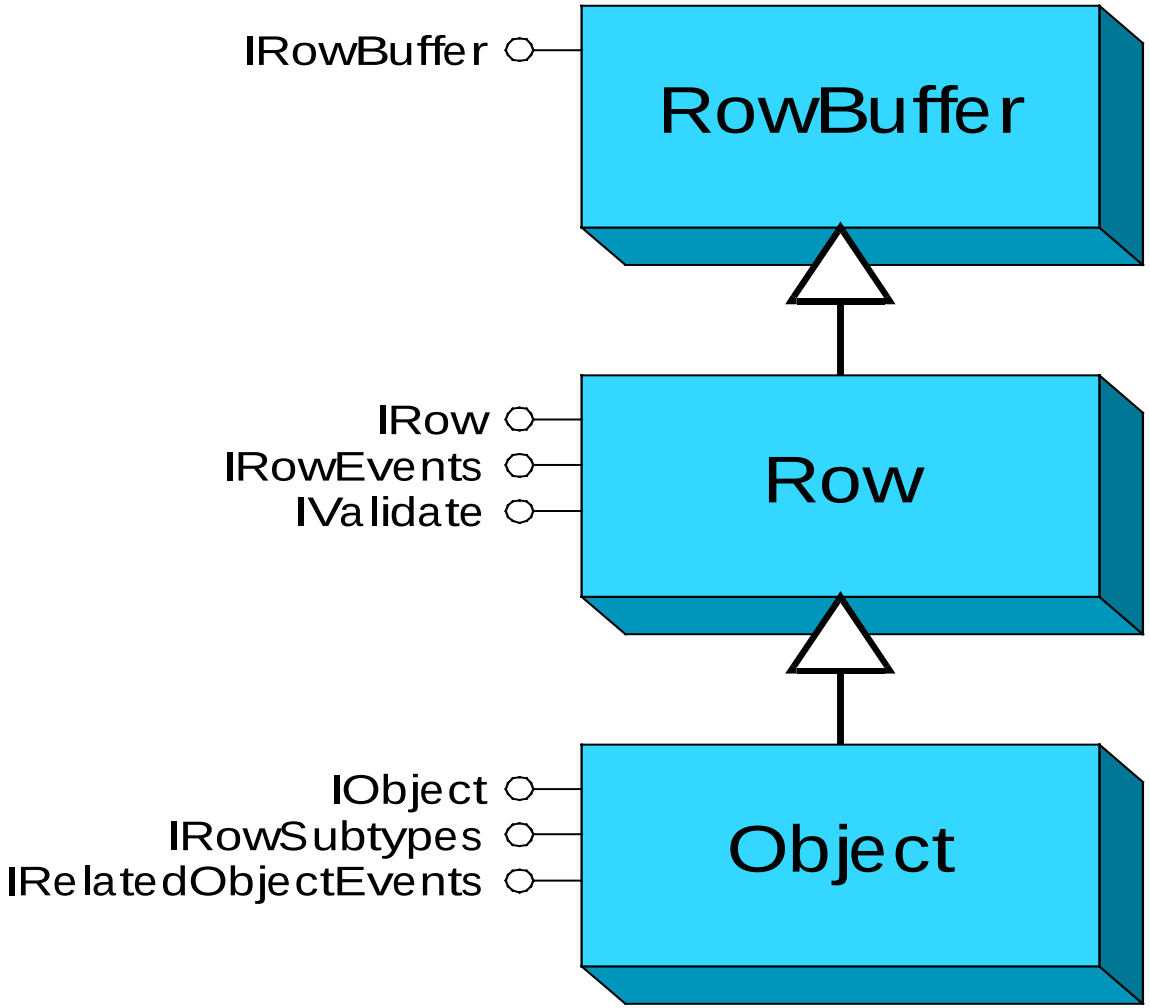
- ***Insert***
  - **Returns an ‘insert’ cursor that can be used to bulk insert rows**
  - ***Significantly* faster performance than multiple calls to ITable.CreateRow for simple data**
- ***Update***
  - **Returns an ‘update’ cursor that can be used to bulk update rows**
  - ***Somewhat* faster performance than multiple calls to IRow.Store for simple data**

ITable

- *Select*
  - Creates a selection set on the table
  - The selection set holds either row objects or object ids
  - Multiple selection sets can be created on the same table
  - A geodatabase table does not have ‘a’ selection (unlike ArcView vtabs)
  - Higher level objects (Feature Layers, Table Windows) etc hold onto selections they create on tables


ISelectionSet : IUnknown
Count: Long
FullName: IName
IDs: IEnumIDs
Target: ITable
Add (in OID: Long)
AddList (in Count: Long, in OIDList:
Combine (in otherSet: ISelectionSet, in setOp: esriSetOperation, out resultSet: ISelectionSet)
MakePermanent
Refresh
RemoveList (in Count: Long, in OIDList: Long)
Search (in pQueryFilter: IQueryFilter, in recycling: Boolean, out ppCursor: ICursor)
Select (in QueryFilter: IQueryFilter, in selType: esriSelectionType, in selOption: esriSelectionOption, in selectionContainer: IWorkspace) : ISelectionSet


IDataset
IDatasetEdit
ISchemaLock
Dataset
IClass
ITable
Table
IClassSchemaEdit
IModelInfo
IObjectClass
ISubtypes
IValidation
ObjectClass


IRowBuffer
RowBuffer
IRow
IRowEvents
IValidate
Row
IObject
IRowSubtypes
IRelatedObjectEvents
Object

ObjectClass

- *ObjectClassId*
  - **Unique within a geodatabase**
  - **Assigned at creation / registration time**
- *RelationshipClasses*
  - **The set of relationship classes in which this object class participates**
- *CLSID*
- *EXTCLSID*

ObjectClass

- **Existing tables not registered with the geodatabase are also presented as object classes.**
- **They have an** *ObjectClassId* **of –1.**
- **They return False to** *IClass::HasOID*

ObjectClass

- IClassSchemaEdit
  - Methods to modify schema information
  - *RegisterAsObjectClass*
    - adds a row to the GDB_ObjectClassesTable
    - adds an object id column if not present
  - *AlterInstanceCLSID*
  - *AlterExtensionCLSID*
  - *AlterAliasName*

SchemaLocks – *Dataset::ISchemaLock*

- Acquire an exclusive schema lock when making any schema modifications to a dataset, eg :
  - Renaming an object
  - Adding a field
  - Changing the feature type of a feature class
    - Eg – building a network
  - Programatically altering the COM behavior class associated with an object class
- Catalog does this when changing schema

ObjectClass

- ISubtypes
  - Enumerate subtypes
  - Get default values and domains by field and subtype
  - Add and delete subtypes
- IValidation
  - Enumerate validation rules
  - Add and delete rules
  - Validate sets of objects

FeatureClass

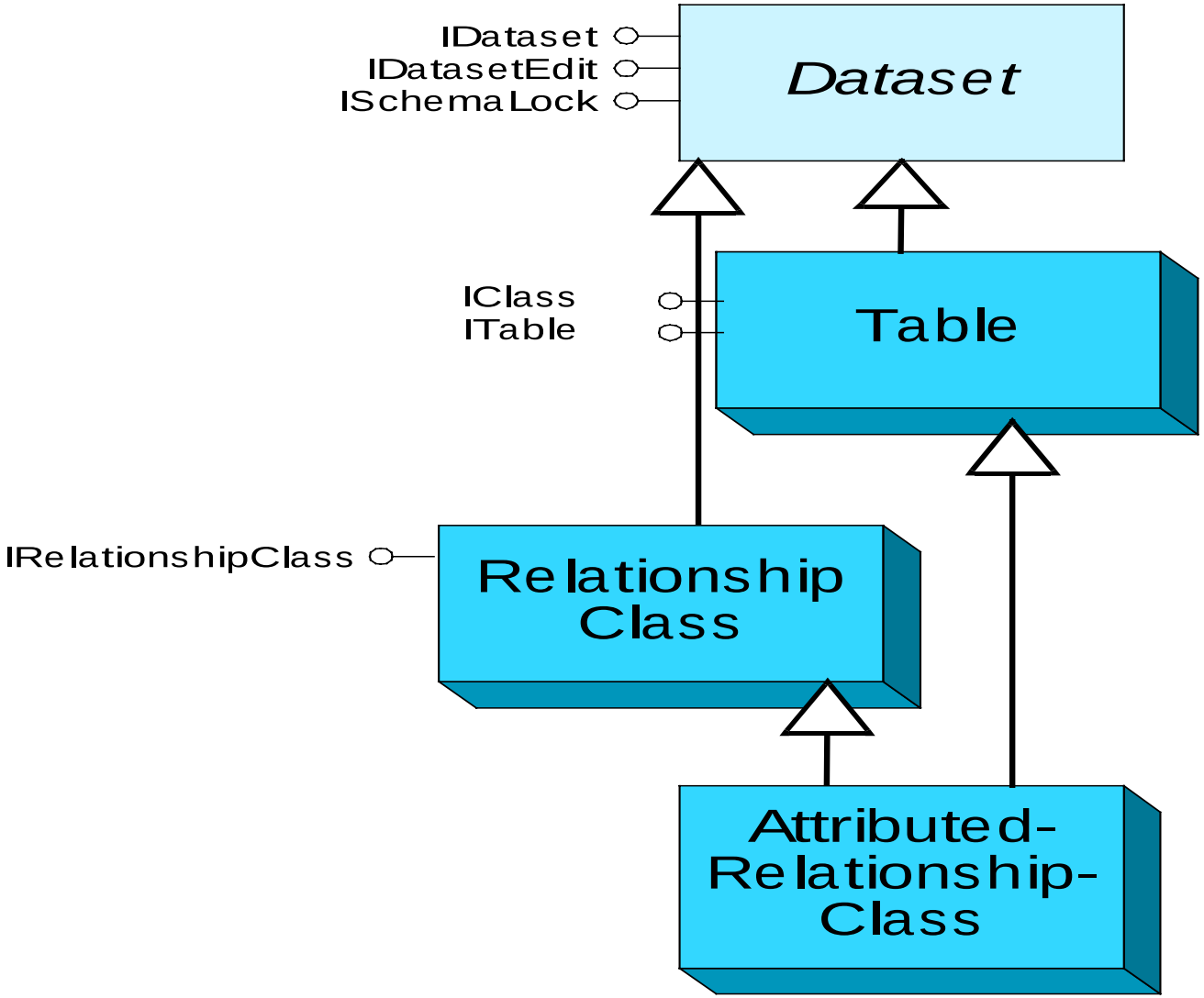
- *FeatureType*
  - **Simple, Network, Annotation, Dimension, …**
- *ShapeFieldName, ShapeType, AreaFieldName, LengthFieldName*
- ***FeatureDataset***
  - **The containing feature dataset, if present**

FeatureClass – API redundancy

- *FeatureClassId*
  - **Same as ObjectClassId**
- ***CreateFeature, GetFeature, GetFeatures***
  - **Same as *ITable CreateRow, GetRow, GetRows*, except return IFeature interfaces on objects**
- ***Search, Insert, Update***
  - **Same as ITable methods, except return IFeatureCursor interfaces on cursors**

RelationshipClass

- Models a relationship between two object classes
- A type of dataset
- Methods to get objects related to other objects
- Methods to add and delete relationships that hide the underlying schema


IDataset
IDatasetEdit
ISchemaLock
Dataset
IClass
ITable
Table
IRelationshipClass
Relationship Class
Attributed-Relationship-Class

IRelationshipClass - properties

- BackwardPathLabel: String
- Cardinality: esriRelCardinality
- DestinationClass: IObjectClass
- DestinationForeignKey: String
- DestinationPrimaryKey: String
- FeatureDataset: IFeatureDataset
- ForwardPathLabel: String
- IsAttributed: Boolean
- IsComposite: Boolean
- Notification: esriRelNotification
- OriginClass: IObjectClass
- OriginForeignKey: String
- OriginPrimaryKey: String
- RelationshipClassID: Long
- RelationshipRules: IEnumRule

IRelationshipClass

- ***GetObjectsRelatedToObject***
  - Takes a single source object
  - Source object may be from the origin or the destination class
- ***GetObjectsRelatedToObjectSet***
  - Takes a set of source objects
  - More efficient then calling GetObjectsRelatedToObject in a loop
- ***GetObjectsMatchingObjectSet***
  - returns a set of matching pairs

IRelationshipClass

- *CreateRelationship*
  - Creates a new relationship between two objects
- *DeleteRelationship*
- *DeleteRelationshipsForObjectSet*
  - Deletes all relationships in which objects in the input set participate

IRelationshipClass

- *GetRelationshipsForObject*
- *GetRelationshipsForObjectSet*
  - Returns an enumeration of relationship objects in which the source objects participate
- Useful for attributed relationships

Edit Sessions

- Object editing is done within an Edit Session
- An Edit Session is a Long Transaction
- The only changes that an editor sees are those made by the editor – takes advantage of the underlying versioned database
- Allows objects like network topology to be safely cached in a multi user environment with introducing concurrency problems (eg. lost updates)

Edit Sessions

- The changes made by an editor are visible to other users only if the edit session commits
- The geodatabase gaurantees ‘unique instancing’ of objects within an edit session.
  - Any data access call that retrieves an object with a particular object id will return the same in memory instance of the object
  - Needed for correctness when updating complex object models – eg. model with relationship based messaging
- Object Editing should be done within an edit session

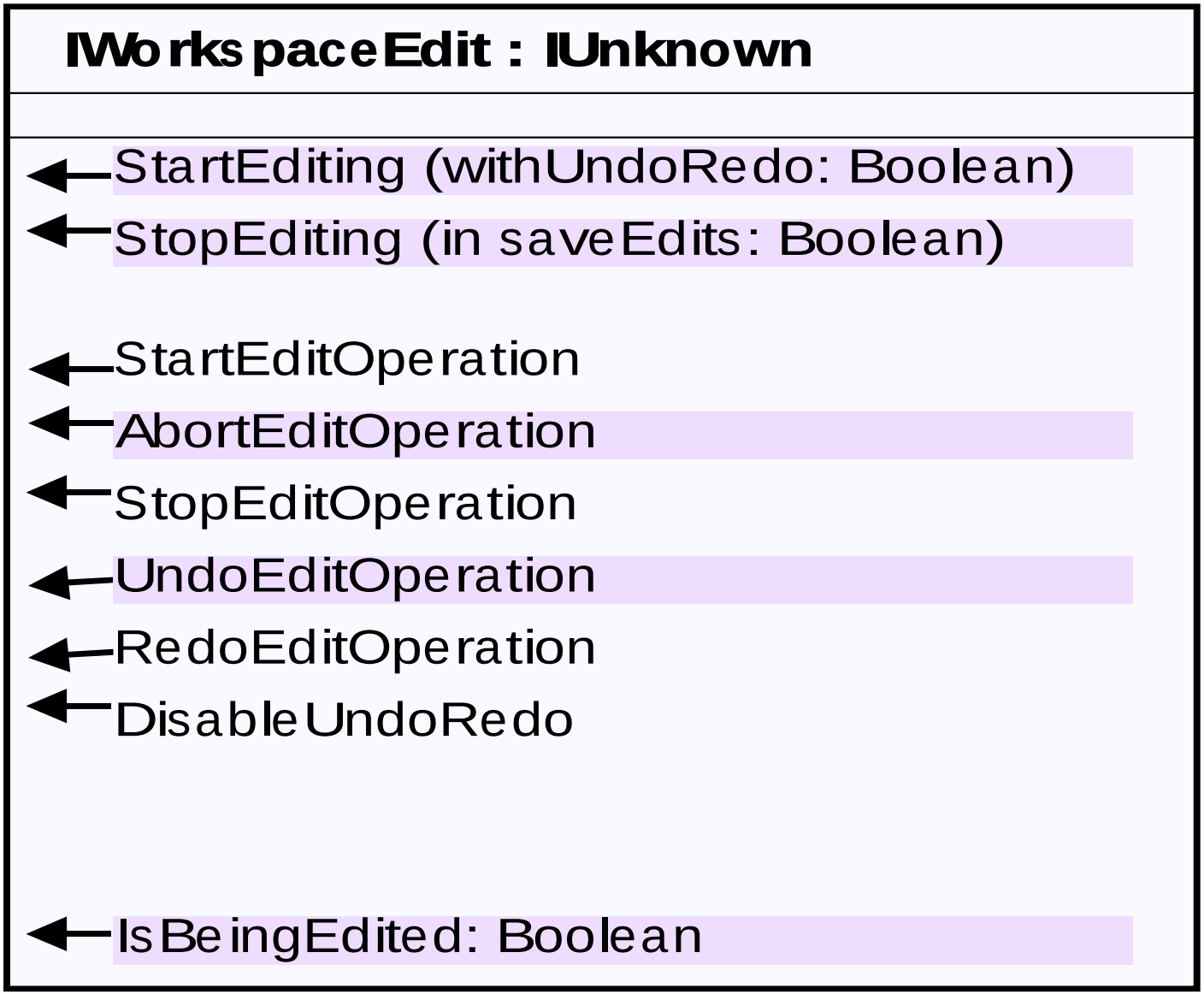
Edit Sessions

- **Data Access update APIs may be used inside or outside of an edit session**
  - **Eg. ITable::Search + IRow::Store**
  - **Eg. ITable::CreateRow, ITable::Insert**

- **Can Use API's outside of an edit session to load and update simple objects and features.**
  - **CLSID_Row, CLSID_Object, CLSID_Feature**
  - **Use database (short) transactions as appropriate**

Edit Sessions

- **Data Access update APIs will return an error if you attempt to use them outside of an edit session on complex objects and features that require an edit session**
  - **Eg. on network features**


IWorkspaceEdit : IUnknown
StartEditing (withUndoRedo: Boolean)
StopEditing (in saveEdits: Boolean)
StartEditOperation
AbortEditOperation
StopEditOperation
UndoEditOperation
RedoEditOperation
DisableUndoRedo
IsBeingEdited: Boolean

Object Messaging

- **Custom Objects can receive and respond to messages from the geodatabase**
  - **Object must implement IRowEvents**
- **Custom Objects can receive and respond to messages from related objects**
  - **Object must implement IRelatedObjectEvents**
  - **Object Class must implement IRelatedObjectClassEvents**

IRowEvents

- *OnInitialize*
  - The object has been hydrated with fresh field values
- *OnNew*
  - Store has been called on this newly created object
- *OnChanged*
  - Store has been called on this existing object
- *OnDelete*
  - Delete has been called on this existing object

IRelatedObjectEvents

- ***RelatedObjectChanged***
  - **The related object has changed**
  - **Use IFeatureChanges to determine if the shape changed, and the original shape**
  - **Use IRowChanges to determine if a specific field value changed and the original value (8.1)**

IRelatedObjectClassEvents

- ***RelatedObjectCreated***
  - **An object was created in a related class**
  - **For eg. an Annotation class is notified when a new feature is created in a related feature class.**

Versioning

- A database may have multiple versions.
- Schema is constant across versions
- Object and database state is different from one version to another :
  - Number of rows in a table
  - Value of a particular column in a particular row
  - The topology at a junction in the network
- Each version is presented to the user as a separate object

Versioning

- An application can access multiple versions simultaneously
- Opening the same table in two versions results in the application holding on to two table objects.
- Allows each table object to cache its own state as needed
- Application can cursor over / issue queries against either table

Version - Interfaces

- IVersion
  - Methods that return information on the version such as its *Name*, *Description*, *Access*, *Parent*, …
- IWorkspace, IFeatureWorspace, …
  - Methods that allow applications to open datasets in the version
- IWorkspaceEdit, IVersionEdit
  - Methods to edit a version and to Reconcile and Post a version within an edit session
- IVersionedWorkspace
  - Methods that support :
    - enumerating all the versions,
    - finding other versions,
    - compressing the versioned database

Conclusion

- Developers have access to a rich set of COM objects and interfaces
- ArcCatalog, ArcMap and the Object Editor use these very same objects and interfaces
- Developers can also use them to write their own application programs
- Programming the Geodatabase is not just custom feature programming.
- Happy Developing !