



Welcome to:

Hitchhiker's Guide to the Geodatabase

Craig Gillgrass

Colin Zwicker



Please!

turn OFF cell phones
and refrain from using
flash photography

Session Path



**Introduction to
Geodatabase
Programming**

Licensing

**Accessing
and Creating
Data**

**Beyond
Basics
And
Editing**

Cursors

**Questions
At end**

**ArcGIS
9.2**

**Extending
The
Geodatabase**

**Conversion
and
Loading**

Session



- Session focus on:
 - How to work with the Geodatabase API
 - As opposed to customizing the Geodatabase
 - Tip\Tricks for using the API
- Lots of content, little time
 - Please hold all questions to the end
 - We'll be at the Tech Talk for questions
 - Also Island area today and tomorrow

Assumptions



- Geodatabase knowledge
 - Basic understanding of Geodatabase concepts and terms
 - programmer/developer or user side
- Basic programming skills
 - C# demos
 - Lots of Code Snippets
 - Ability to read OMDs
 - ArcObjects supports: VB 6, .Net, Java, etc

Session Path



**Introduction to
Geodatabase
Programming**

Licensing

**Accessing
and Creating
Data**

**Beyond
Basics
And
Editing**

Cursors

**Questions
At end**

**ArcGIS
9.2**

**Extending
The
Geodatabase**

**Conversion
and
Loading**

Geodatabase



- Geodatabase is...
 - Core ArcGIS data model
 - Set of components in ArcGIS for accessing data
 - A physical store of geographic data
- Built on Relational DBMS
- Geodatabase is a workspace
 - Contains datasets; feature classes, tables, etc
 - Extend with Topologies, Networks, etc



Geodatabase terms

- Geodatabase types
 - Personal geodatabase
 - File Geodatabase (ArcGIS 9.2)
 - Personal, Workgroup and Enterprise ArcSDE geodatabases
 - Geodatabase XML
- Geodatabase API
 - Building blocks for programming with GIS data
 - Not just the native Geodatabase model
 - Leverage the API against other dataset
 - Shapefiles
 - CAD
 - Coverage
 - etc

Geodatabase terms



- Benefits of the Geodatabase
 - High level design
 - Integrity tools
 - Leverage business rules
 - Rich editing model, support of long transactions
 - Variety of containers

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

Questions
At end

ArcGIS
9.2

Extending
The
Geodatabase

Conversion
and
Loading

Licensing



- Required to Initialize licensing to correct level when working with Geodatabase in Engine environment
 - If not, get an error on call to open the workspace
- Configure license at application start time
 - `esriLicenseProductCodeEngineGeoDB`
- Enterprise geodatabase editing application requires any of:
 - ArcGIS Engine GeoDatabase Editing license
 - ArcEditor license
 - ArcInfo license

Session Path



Introduction to
Geodatabase
Programming

Licensing

**Accessing
and Creating
Data**

Beyond
Basics
And
Editing

Cursors

**Questions
At end**

**ArcGIS
9.2**

**Extending
The
Geodatabase**

**Conversion
and
Loading**



Workspace Factory and Workspaces

- Dispenser of workspaces
 - Personal, SDE, etc
- Create a factory from a factory coclass
- *A workspace is a container of datasets.*
- Examples :
 - Geodatabase, Coverage Workspace, Folder of Shapefiles
- Create a workspace from a factory
 - Path to data and window handle (app ID)
 - For SDE use .sde connection file or propertyset
 - For string use connection properties (next)

IWorkspaceFactory ○

WorkspaceFactory

IWorkspaceFactory2 ○

Workspace



Opening datasets

- Dataset model
 - Open methods on IFeatureWorkspace, IFeatureClassContainer, etc
- Dataset Extensibility model
 - Open methods on IDatasetContainer2
- Use a name – "Streets"
 - Use ISQLSyntax::QualifyTableName for SDE
 - Owner not required to qualify name
- Use a ClassID
 - ClassIDs are unique and sequential within Geodatabase
- Other methods; Index, Enumerations, etc

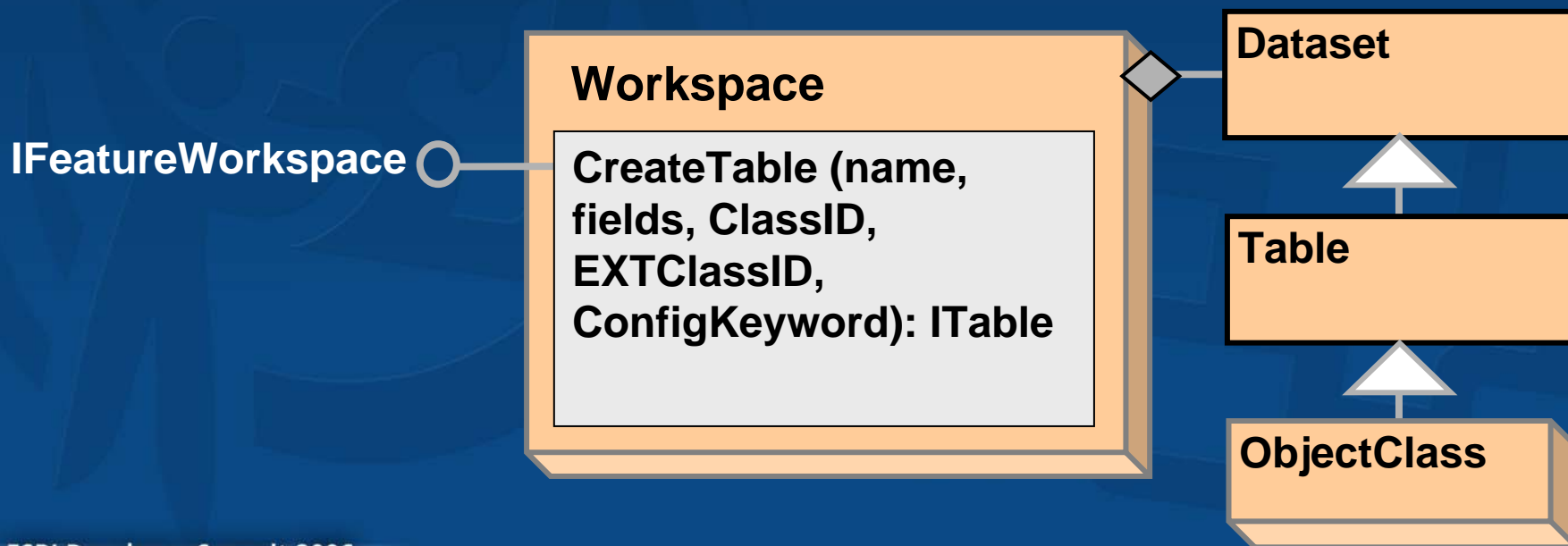
Creating data



- Create methods to match the open methods
 - FeatureClass
 - FeatureDataset
 - Table
 - ...
- Dataset model
 - Properties of dataset to be created arguments to Create method
- Dataset Extensibility model
 - Use a Data Element that has been pre-populated
 - Only supports Network Datasets at 9.1

Create a table

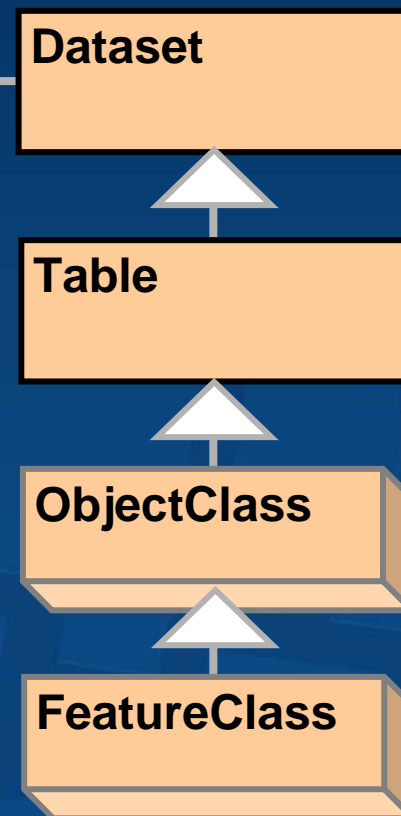
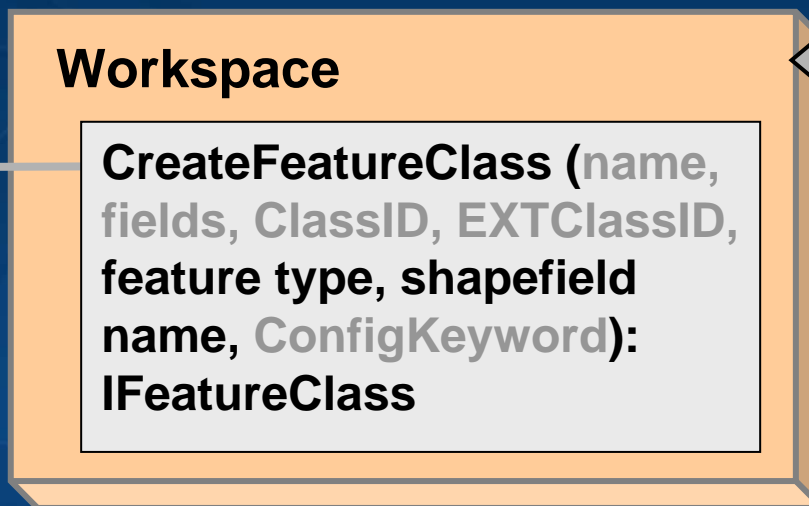
- Creates an ObjectClass, returns ITable interface
 - ObjectID field. Values are Never Reused.
- Custom behavior ID's (*can use null*)
- SDE configuration keyword (*can use " "*)
- Can create Fields first
 - Use IObjectClassDescription:RequiredFields



Create a Feature Class

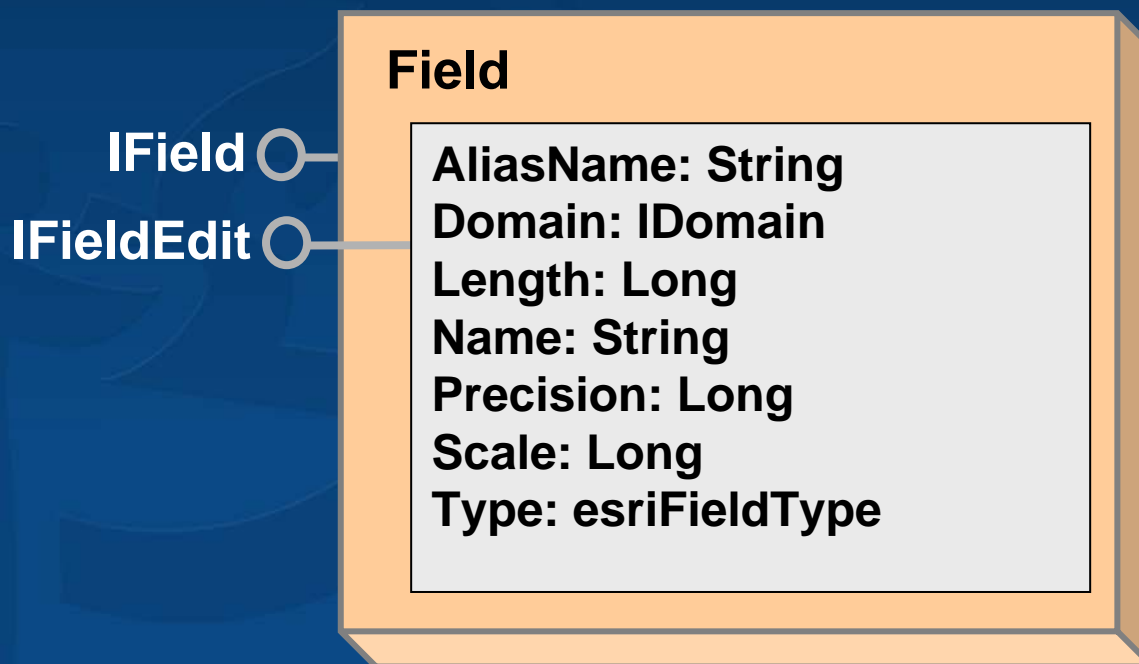
- Same as CreateTable except:
 - Needs Shape type and Shape field name
 - IGeometryDefEdit used when defining new feature class
 - Use IFeatureClassDescription:RequiredFields
 - Returns IFeatureClass interface

IFeatureWorkspace



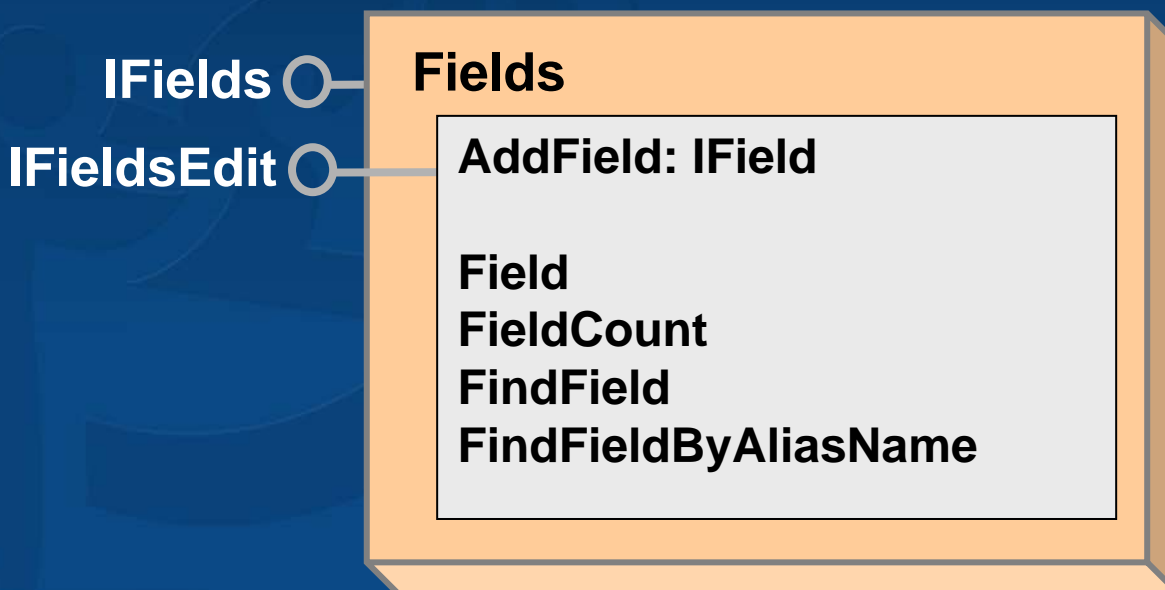
Create a Field

- Set properties with IFieldEdit
- Types include: Integer, Single, Double, String, Date, OID, Geometry, Blob, GUID, and Raster
- Geodatabase tables must have an OID field
 - Using Class Descriptions will take care of this



Create a Field cont'd

- Use a Field collection (Fields) when creating datasets
- Adding fields
 - **Only** use Fields object for initial creation
 - For existing tables use IClass::AddField method to add fields
- Set properties for the Field with the IFieldEdit interface
- Leverage Class Description whenever possible
 - ObjectClassDescription, FeatureClassDescription, etc



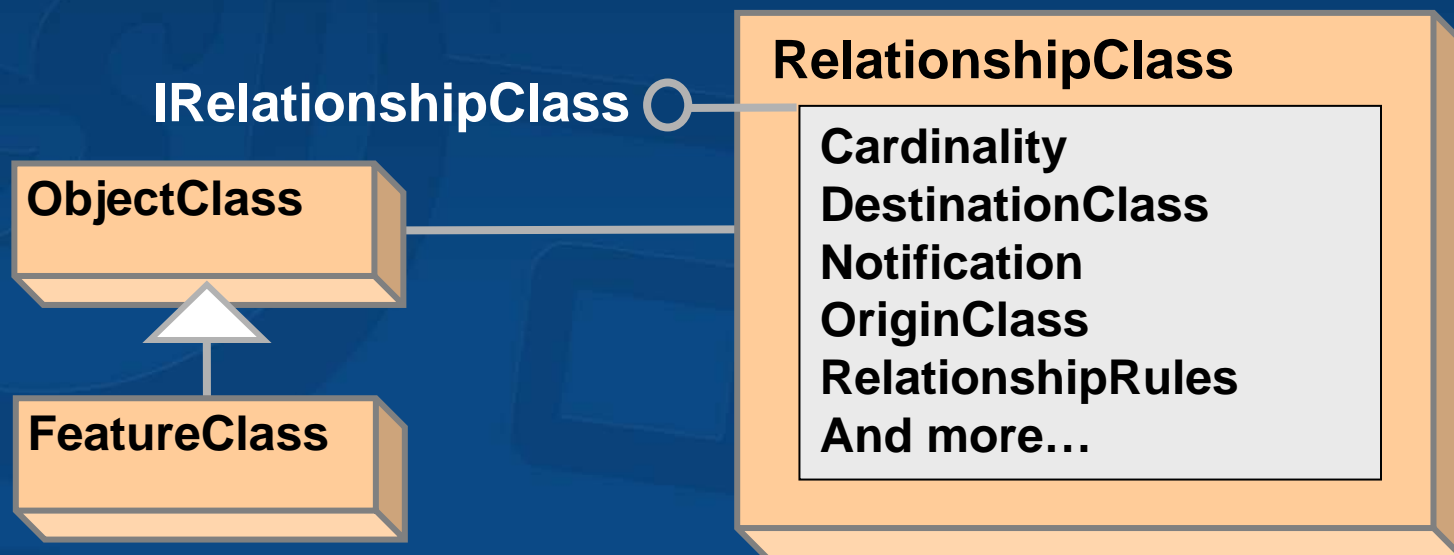


Domains and Subtypes

- Domain created at workspace level
 - Domains constrain field values
 - Associate a Domain to a field(s)
 - During create use IFieldEdit:Domain
 - Or IClassSchemaEdit:AlterDomain
- Subtypes are specific to feature class
 - Partition the objects in a class into like groups
 - Defined by the value of a subtype field
 - Can constrain rules\behavior to the Subtype level
 - Have a default subtype code
 - Important for feature creation and editing

RelationshipClass Properties

- Create with IFeatureWorkspace or IRelationshipClassContainer
- You set properties to define the relationship
 - Origin and destination tables
 - Primary – Foreign key relations (fields)
 - RelationshipRules, Notification, Cardinality
- Find related objects
 - Get related object sets using GetObject methods



Data Access and Creation demo



- Create a Personal Geodatabase (PGDB)
- Create a feature dataset and feature class
- Creating Domains, Subtypes and Rules



Create rows and features

- Basic process to create row or feature
 - CreateRow or CreateFeature
 - Can also use InsertCursor, more later
 - If subtypes present, set IRowSubtypes::SubtypeCode
 - If default values, call IRowSubtypes::InitDefaultValues
 - Set attribute values
 - Create geometry and set Shape
 - Call Store
 - writes the values to the row in the table



Create rows and features

- Non-simple feature creation must be in an edit session\edit operation
- Any dataset specific behavior; ie: for features created in Geometric Networks, Topologies, etc; is handled at creation time
 - Not required to call Connect or create Dirty Areas

Data Access and Creation demo



- Creating features
- Set values for the new feature
- Store values

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

**Beyond
Basics
And
Editing**

Cursors

**Questions
At end**

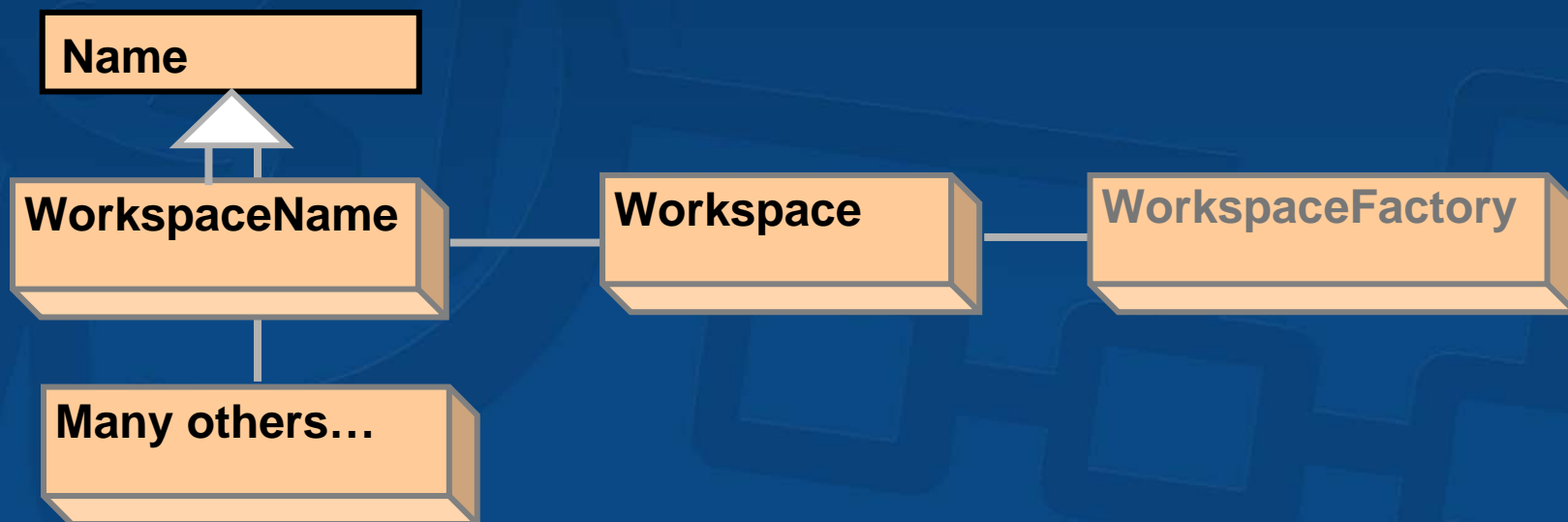
**ArcGIS
9.2**

**Extending
The
Geodatabase**

**Conversion
and
Loading**

Name objects

- Light-weight representation of real object
 - Name objects for each dataset
 - Acts as a moniker to the dataset
- Can be persisted
- Allows you to bind to the real object
 - You can get some properties from the name object





DataElement objects

- Used in geoprocessing functions, web services
Dataset Extensibility
 - Describe actual geodatabase datasets
 - DEFolder, DETable, and DEShapeFile, etc
- Simple structures whose properties describe the actual entity
 - Different from Name objects, you cannot directly open the dataset
- Support *IXMLSerialize* and *IPersistStream*
 - can be serialized in XML or binary form



RelQueryTable (Joins) vs QueryDefs

- Both are used to return results from 2 tables
 - But for different reasons
- RelQueryTable
 - Joins two datasets
 - Can be across different data sources
 - Treat like a Table or FeatureClass
 - Based on a RelationshipClass or MemoryRelationshipClass
 - Use IDisplayTable in ArcMap (esriCarto.olb)
- QueryDefs
 - Query based on one or more tables
 - Analogous to an SQL Query
 - Get a Cursor back
 - Tables must be in database
 - Do not have to be registered
 - Can result in a feature layer
 - Need to use TableQueryName::IQueryName2 if no ObjectIDs in input tables

Geometric Networks



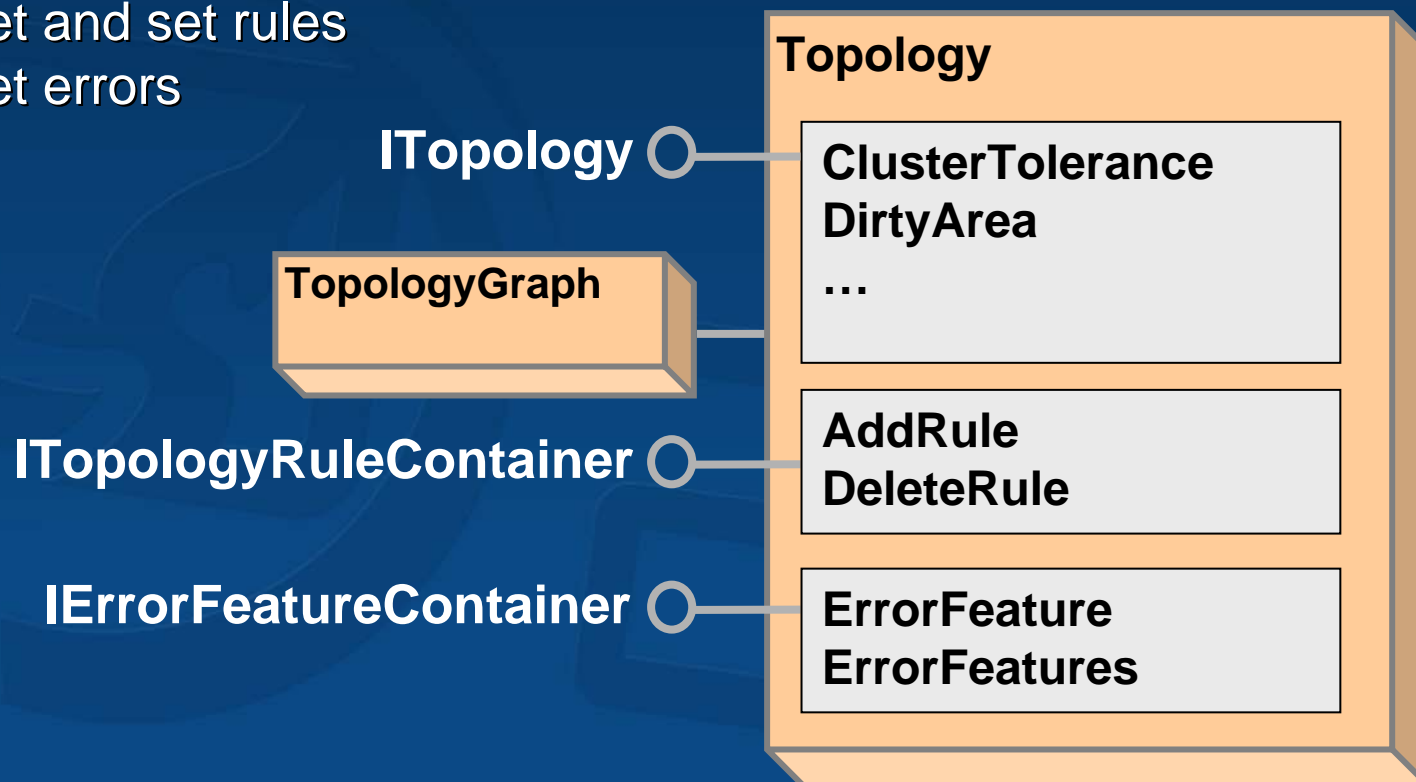
- Designed for Utilities\Natural Resources industries
- Use INetworkLoader for creation of Geometric Networks
- Use Logical Network API for navigation and tracing whenever possible
 - IForwardStar
- Navigational APIs available at the geometric network feature level
 - very slow
 - use only for small tactical navigation
- Analysis algorithms (e.g., solvers) should always consume the logical network APIs
 - orders of magnitude faster
 - INetwork
 - INetTopology

Network Datasets

- Designed for Transportation industry
- Dataset Extension
 - Leverages Data Element and IDatasetContainer2 for creation and updates
- Different than Geometric Networks
 - Simple features
 - No custom behavior
 - Performance is realized with InsertCursors
- INetworkForwardStar and INetworkForwardStartAdjacencies should be used for connectivity analysis
- Support Custom Evaluators (i.e. Cost, Restrictions) and Custom Solvers

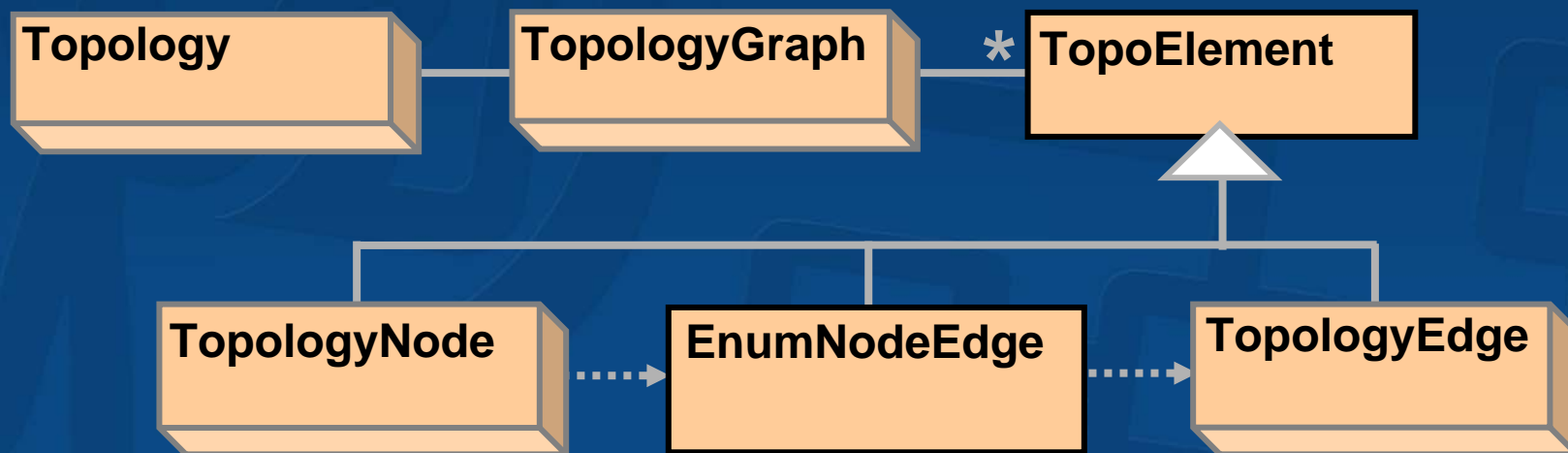
Topology interfaces

- A feature dataset can contain many topologies
- A topology can contain many feature classes and many topology error features
 - Validate topology
 - Get and set rules
 - Get errors



Topology graph and elements

- Each topology has a graph of elements
 - Planar representation
- Use ITopologyGraph to
 - Edit participating topology features
 - Without breaking adjacency or topological relationships
 - Graph access to topo relationships between features



Geodatabase Editing - Edit Session



- Geodatabase explicitly stores change information when edited
- Only see the change you've made within the edit session
 - Changes made by other applications are not seen
 - Until Save or Discard
- Edits should be made within an edit operation
 - StartEditOperation – StopEditOperation
- Each edit operation represents a transaction
 - **Stop** commits the change
 - **Abort** rolls back, like undo
- Applications are responsible for calling:
 - AbortEditOperation when errors are detected
 - StopEditOperation to complete edit operations
 - Pushed to the undo stack
- UndoEditOperation, RedoEditOperation
 - Geodatabase moves the operation between the Undo and Redo stacks

Editing the Geodatabase

- When to use Edit Sessions?
 - Always...
 - Must use with topologies, geometric networks, annotation, relationship classes
 - Use `IObjectClassInfo2::CanBypassEditSession`
- When to use `IEditor` or `IWorkspaceEdit`?
 - Use `IEditor` to edit within an application, like ArcMap
 - Ensures Undo/Redo consistency between edits made programmatically and through the UI
 - Must use `IWorkspaceEdit` in Engine environment
- Similar methods on each





Editing Demo

Colin

- List Domain values for feature based on Subtype
- Topological Editing

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

Questions
At end

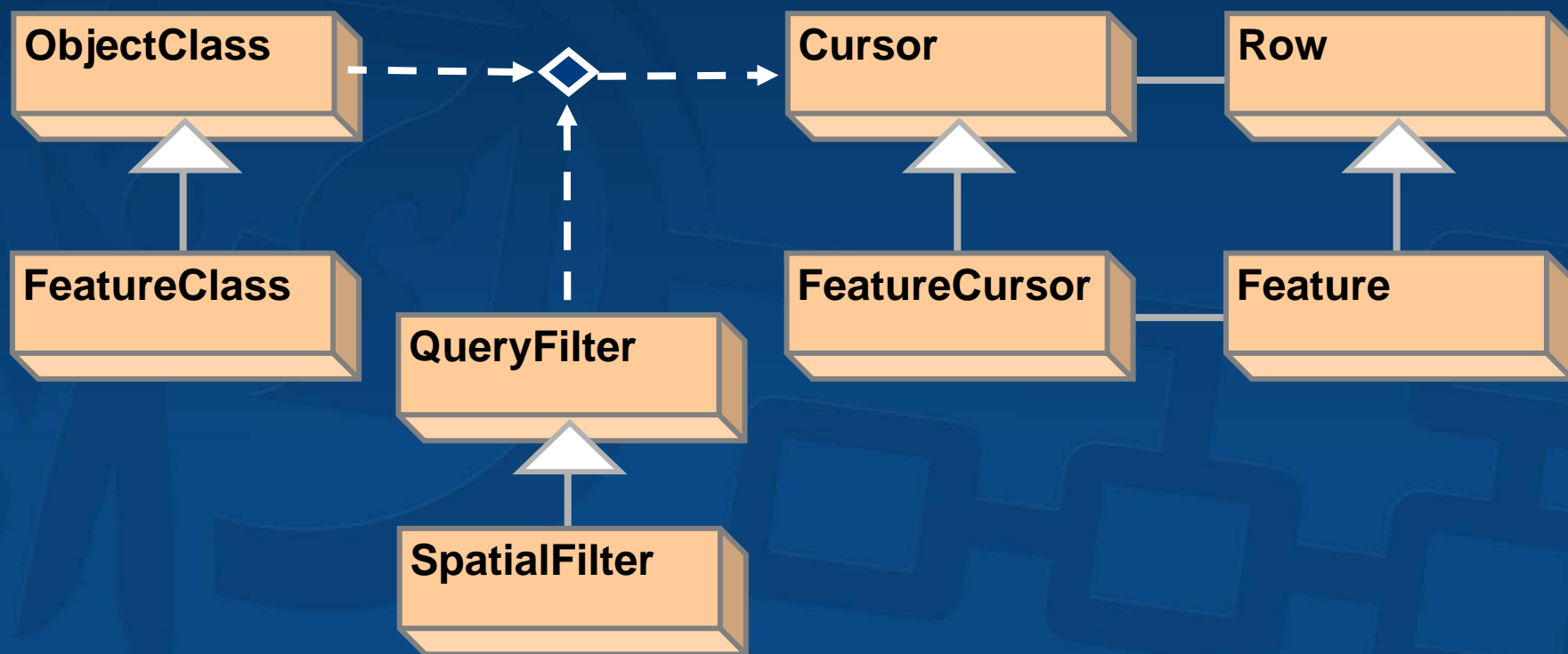
ArcGIS
9.2

Extending
The
Geodatabase

Conversion
and
Loading

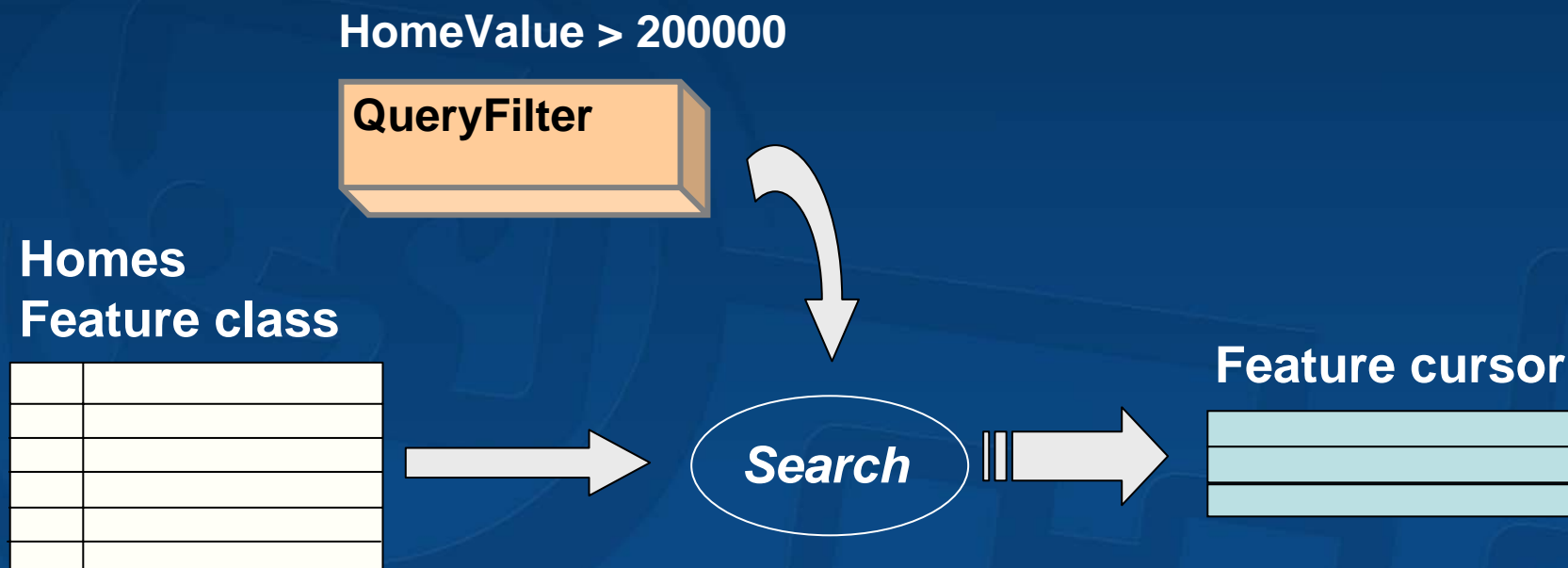
Cursors

- A table and a query return a cursor
- Used to:
 - Iterate over a set of rows in a table
 - Insert new rows into a table
- A cursor gives you access to one row at a time



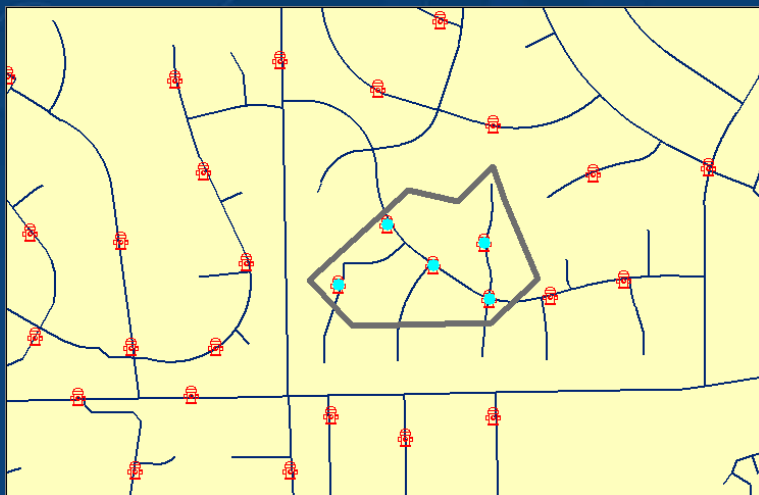
Creating a cursor

- QueryFilter contains an SQL-like statement
- The cursor contains a subset
 - No filter\Nothing, all rows returned



Creating a cursor

- SpatialFilter need a geometry and relationship
- Below the geometry is a polygon
- Below the spatial relationship is *contains*



Contains
Crosses
Intersects
Overlaps
Touches
Within



3 Types of Cursor

- Search cursors
 - Returns rows specified by a Query or Spatial Filter
- Update cursors
 - Update and Delete rows specified by the QueryFilter
 - Specify the ObjectID field
- Insert cursors
 - Used for inserting rows into a table
- Accessed by
 - Corresponding methods on Table or FeatureClass
- Forward only, do not support
 - Backing up and retrieving rows already retrieved
 - Making multiple passes
 - Reseting
 - Solution: Re-execute the query
- Release FeatureCursors with `Marshal.ReleaseComObject`



Recycling Method

```
pCursor = theMeds.Update(pFilter, false)
```

Recycling

- Allocate a single row object
 - Re-hydrate on each fetch
- Performance advantages
- Should only be used for reading data
- Non Recycling
 - A different row object on each fetch
 - Always has full set of fields, even if IQueryFilter::Subfields used
- Not sure what to do?
 - Use a non recycling cursor

InsertCursor vs ObjectLoader

- Insert cursors are used to bulk insert rows
 - Faster for loading simple data
 - By passes events
 - IObjectClassInfo2 and IWorkspaceEditControl to override
 - Not Faster for Non-simple data
 - Behavior, composite relationships, and notification
 - Need CreateRow and Store methods, so no performance gain
 - Use of Buffering is key
 - Pre-define attribute values
 - Buffers inserts on client, sends to DB on Flush
- Flush – Call or not
 - Interval flushing: Check for room or handle errors
 - Careful: Insert Cursors flush on destruction
 - No chance to detect errors
- ObjectLoader provides more control
 - Field checking
 - Progress checking
 - List of invalid objects



Fields with Cursors, Tables and Rows

- All row objects retrieved from a Table using a Cursor, logically contain the same ordered set of fields
- Index of field remains consistent
 - Index of a field in the Fields collection of a table
 - Index of the field in the Fields collection of the cursor
 - Index of the field in the Fields collection for the row
- IFields:FindField
 - Equivalent methods on IClass, ICursor



Cursor demo

Colin

- Cursors examples
 - Insert
 - Update
 - Search

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

Questions
At end

ArcGIS
9.2

Extending
The
Geodatabase

Conversion
and
Loading



Conversion and Loading

- Loading into the Geodatabase
 - IFeatureDataConverter
 - Used for converting simple features only
 - A lot of set-up required
 - Geoprocessing tools
 - Course grained, facilitates loading
- Between Geodatabases
 - IGeoDBDataTransfer
 - Supports non-simple features
 - Works at Dataset level
 - IGdbXmlImport and IGdbXmlExport
 - Supports non-simple features
 - Workspace\Dataset level
 - Can also use Geoprocessing tools



Conversion Demo

Colin

- Using FeatureDataConverter vs Geoprocessing tool to load data

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

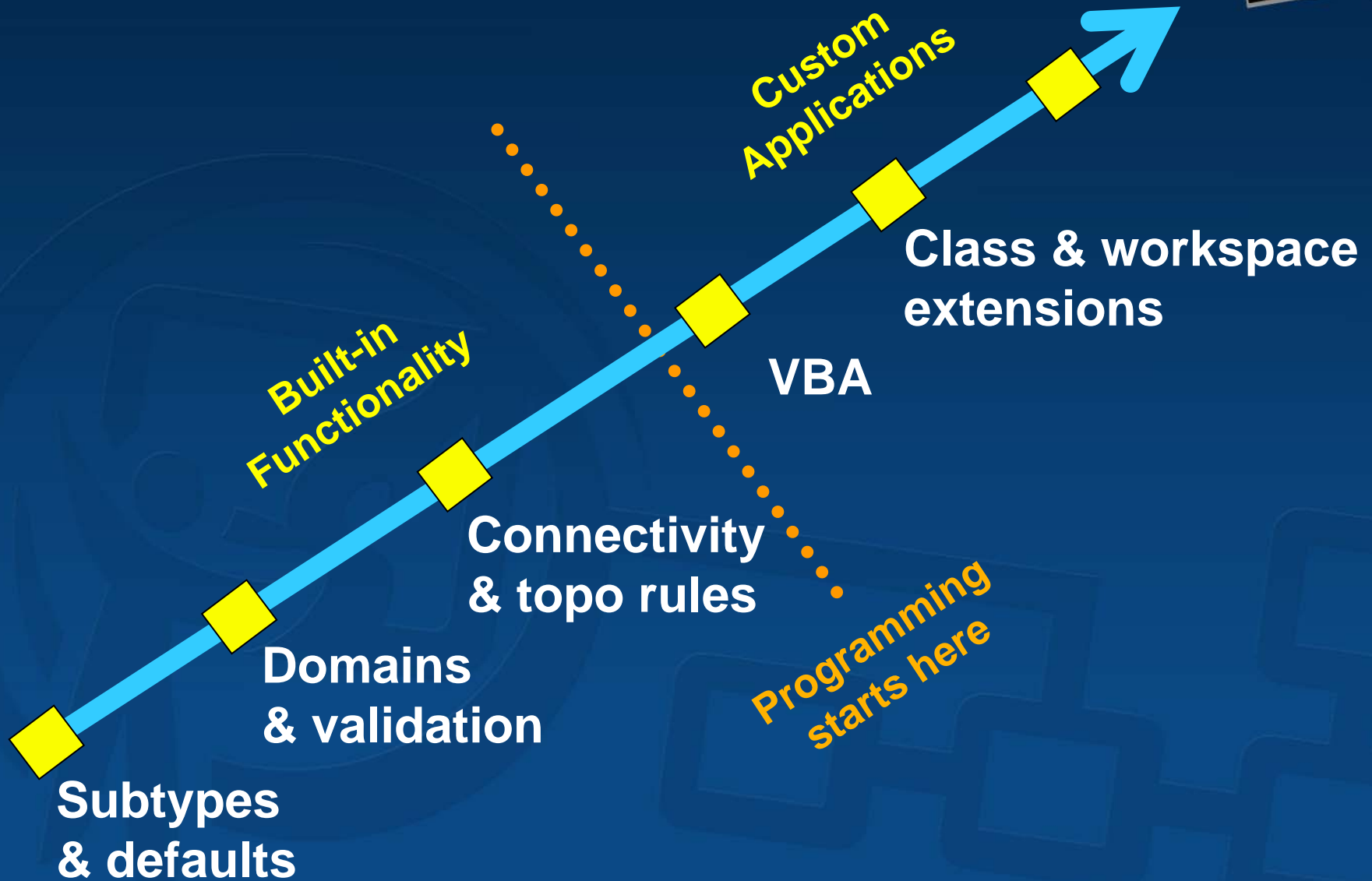
Questions
At end

ArcGIS
9.2

**Extending
The
Geodatabase**

Conversion
and
Loading

Levels of customization



Level of Customization



- Application level
 - Pros
 - Business Logic is stored within application
 - Can access data without customization
 - Cons
 - Only available when application is running
 - Users do not always interact with application customization

Level of Customization



- Database level
 - Pros
 - Business Logic is stored with data
 - Always available, regardless of application
 - Cons
 - One class extension per feature class
 - All users require dll to even view data
 - Database is unusable if code fails
 - Impacts on performance
- Use for important business rules that can be simply implemented without serious performance considerations.

Level of Customization



- Custom Features
 - Pros
 - Provide near total control over functionality.
 - Cons
 - Performance can suffer, since code is executed redundantly for every feature.
 - Handling of row and relationship events is less stable than class extensions.
 - Technically challenging to implement.
- Problem can often be solved by class extension or application customization



Class extension uses

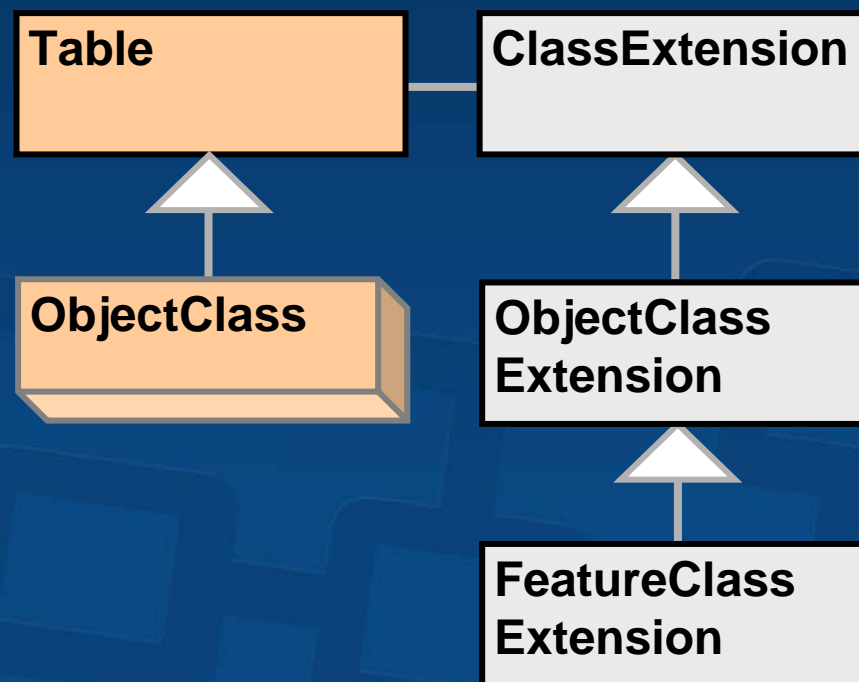
- Schema generation
- Custom drawing
- Custom property inspection and validation
- Custom split policies
- Related object creation notification
- PlugIn Data Sources

Framework interfaces



- **IClassExtension**
 - Used to implement custom behavior for an objectclass
 - In Init, keep a reference to ClassHelper, not Class itself

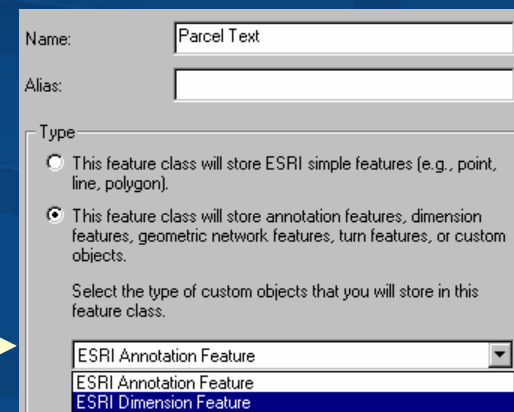
- Indicator interfaces
- No Members
 - IObjectClassExtension
 - IFeatureClassExtension



Customization interfaces

- IFeatureClassDraw
 - Override a feature class's drawing for any client
 - Consider: a custom renderer
- Add to editing experience
 - IObjectInspector
 - IFeatureClassEdit
- Control new class creation
 - IObjectClassDescription
 - IFeatureClassDescription
 - INetworkClassDescription

Your description here →
(In ArcCatalog)



Name: Parcel Text

Alias:

Type

This feature class will store ESRI simple features (e.g., point, line, polygon).

This feature class will store annotation features, dimension features, geometric network features, turn features, or custom objects.

Select the type of custom objects that you will store in this feature class.

ESRI Annotation Feature

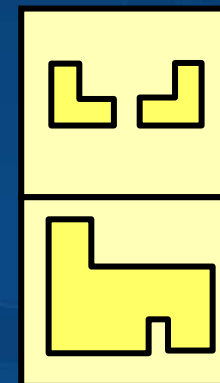
ESRI Annotation Feature

ESRI Dimension Feature

Behavior interfaces

- IObjectClassValidation
- IObjectClassEvents (*OnCreate, OnDelete, OnChange*)
- IRelatedObjectClassEvents
- IRelatedObjectClassEvents2
- IConfirmSendRelatedObjectEvents
- ITopologyClassEvents

**Validation event:
Building height
must be 10 times
the number of stories**

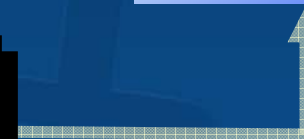
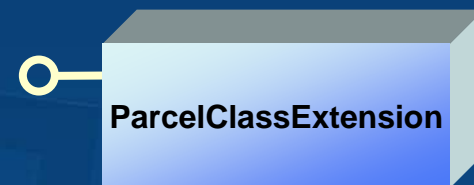


Register and Associate

- Component requires registration
 - On each client system
 - In categories: ESRI GeoObject Class Extension
- GDB associates class extension GUID with class
 - During data creation: IFeatureWorkspace
 - IClassSchemaEdit.AlterClassExtensionCLSID
 - IClassSchemaEdit2.AlterClassExtensionProperties
 - IFeatureWorkspaceSchemaEdit.AlterClassExtensionCLSID

Feature class table within a geodatabase

| ID | Name | CLSID | EXTCLSID |
|----|-----------|-------------|--------------|
| 1 | Parcels | {3070721... | {0368CF51... |
| 2 | Buildings | {3070721... | |
| 3 | Owners | {3070721... | {044782D... |



Class Extensions Demo

- FeatureClassDescription – Predefined feature class

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

Questions
At end

**ArcGIS
9.2**

Extending
The
Geodatabase

Conversion
and
Loading

Key Geodata Management Themes for 9.2



- More Data Sources and Improved Administration
 - Bundled Database Servers
 - Mobile Field operations
 - Disconnected editors and database replicas
 - Small government offices\organizations
 - Easy to install and use
- High Performance File Geodatabase
 - Support for very large data
 - Cross platform support

Key Geodata Management Themes for 9.2



- Better Integration with IT systems
 - Non-versioned Editing
 - Facilitates integration
 - SQL Type for SDE
 - Access to spatial information through SQL
- Enhanced Transaction Model
 - History
 - Versioned Data Replication
 - Reconciliation Enhancements
- Improved Vector Data Management
 - High Precision Data Storage
 - Improved Spatial Processing
- Improved Raster Data Management
- Support for very large Terrains (continuous surfaces)



ArcGIS 9.2 demo

- File Geodatabase creation

Session Path



Introduction to
Geodatabase
Programming

Licensing

Accessing
and Creating
Data

Beyond
Basics
And
Editing

Cursors

**Questions
At end**

ArcGIS
9.2

Extending
The
Geodatabase

Conversion
and
Loading



Other sessions

- **Implementing Enterprise Applications with the geodatabase and ArcSDE**
 - *2:30 - SmokeTree A & B*
- **Leveraging ESRI Data Models in Your Custom Solution**
 - *2:30 - SmokeTree F*
- **Hitchhiker's Guide to Geoprocessing**
 - *4:00 - SmokeTree F*
- **Hitchhiker's Guide to Projections and Spatial References**
 - *Saturday - 2:30 - SmokeTree F*
- **Developing with the ArcGIS Cartographic APIs**
 - *Saturday - 4:00 - SmokeTree A & B*

Conclusion



- Thank you for coming!
- Any other questions?
 - Tech Talks