

# Esri Developer Summit

March 26–29, 2012 | Palm Springs, California

[esri.com/events/devsummit](http://esri.com/events/devsummit)



## Developer's Guide to Geodata Access in ArcGIS

Craig Gillgrass, Brent Pierce, Gary MacDougall, Colin Zwicker



# Schedule

- 3 hours with break from 2:30 pm to 3:00 pm
- Cell phones and pagers

*Please!*  
Turn *OFF* cell phones  
and paging devices



- Please complete the **session survey** – we take your feedback very seriously!

# Presentation Outline

- **Introduction**
- **Databases and Geodatabases**
- **SQL**
- **ArcObjects**
- **Plug In Data Sources**
- **Python**
- **File Geodatabase API**
- **Runtime**
- **Web Editing**

# Seminar Roadmap

- **Assume you have a basic understanding of the geodatabase and databases**
- **Basic programming skills**
  - **Demos in various languages**
    - **C#, Python, SQL, JavaScript, etc**

## **Seminar Roadmap ...**

---

- **We'll take questions throughout**
  - **May want to hold off until we get to the end of each topic**
  - **Hold questions to the end of demos**
- **Available at the break**
- **Showcase area over the next few days**

# ArcGIS 10 — A Complete System

Easier  
More Powerful  
and Everywhere



# Presentation Outline

- Introduction
- **Databases and Geodatabases**
- SQL
- ArcObjects
- Plug In Data Sources
- Python
- File Geodatabase API
- Runtime
- Web Editing

# Databases

- **You might have spatial or nonspatial data in a database that you want to use in ArcGIS**
  - **Oracle, SQL Server, DB2, Informix, PostGreSQL, Netezza**
- **You can connect directly to a supported database and view the data in the tables by making a connection from the Catalog tree in ArcGIS for Desktop**
- **To filter what data appears in ArcMap, you can use a query layer**
- **Use SQL to access the data within the database**



# What can you access in a Database?

- Rows and Tables
  - Containing zero to many rows
  - One to many columns
  - All rows in the table have the same schema
- Can perform table management tasks
  - View and modify schema
  - Add and remove rows
  - Perform queries



The screenshot shows a window titled "Attributes of Riverside\_History.DBO.OWNERS". It displays a table with the following columns: RP, APPR\_YEAR, ACCOUNT, RCD\_TYPE, SEQNO, SALE\_DATE, DEED\_BOOK, DEED\_PAGE, SALE\_PRICE, GRANTOR, GRANTEE, and VACA. The table contains six rows of data representing property sales. Below the table, there is a navigation bar with "Record:" followed by a scroll bar showing "0", a "Show:" dropdown menu set to "All Selected", and a status indicator "Records (0 out of 846 Selected)".

RP	APPR_YEAR	ACCOUNT	RCD_TYPE	SEQNO	SALE_DATE	DEED_BOOK	DEED_PAGE	SALE_PRICE	GRANTOR	GRANTEE	VACA
R	2002	00562424	SALE	000	2/17/2000	14243	247	30000	Frieling, D Rynn	Fullwood, Troy	
R	2002	00565904	SALE	000	4/7/2000	14313	241	47500	Keele, Leslie D	Honell, Bobby & Ivy R	
R	2002	00565717	SALE	000	6/5/2000	14375	190	56752	Hudson, Richard J	Brant, Dorothy Laureline	
R	2002	00566270	SALE	000	7/11/2000	14429	206	65000	Wills, Mary E	Copeland, Jerry Don	
R	2002	00567655	SALE	000	5/9/2000	14300	137	65000	Purcell, Charity F	Churkey, Dale Eluz Johanna	
R	2002	00568066	SALE	000	5/23/2000	14356	330	145000	Hawkins, Gary D Eluz Deborah L	Kjeldgaard, Larry Eluz Linda M	

# What can you access in a Database? ...

- A table with a column that stores a spatial type
  - We call this a feature class
- Each row represents a feature
- The fields in each row represent various characteristics or properties of the feature
- One of the fields holds the feature geometry which is stored as a spatial type

OBJECTID *	SHAPE *	PROPERTY_I *	Res	Zoning_simple	SHAPE_Length	SHAPE_Area
1	Polygon	5001	Non-Residential	<Null>	3597.780813	112552.418591
2	Polygon	5002	Non-Residential	<Null>	814.855837	18488.417709
3	Polygon	1003	Residential	Residential	489.855523	12815.591379
4	Polygon	1004	Residential	Residential	521.761240	14036.135346
5	Polygon	1005	Residential	Residential	453.479649	9016.352665



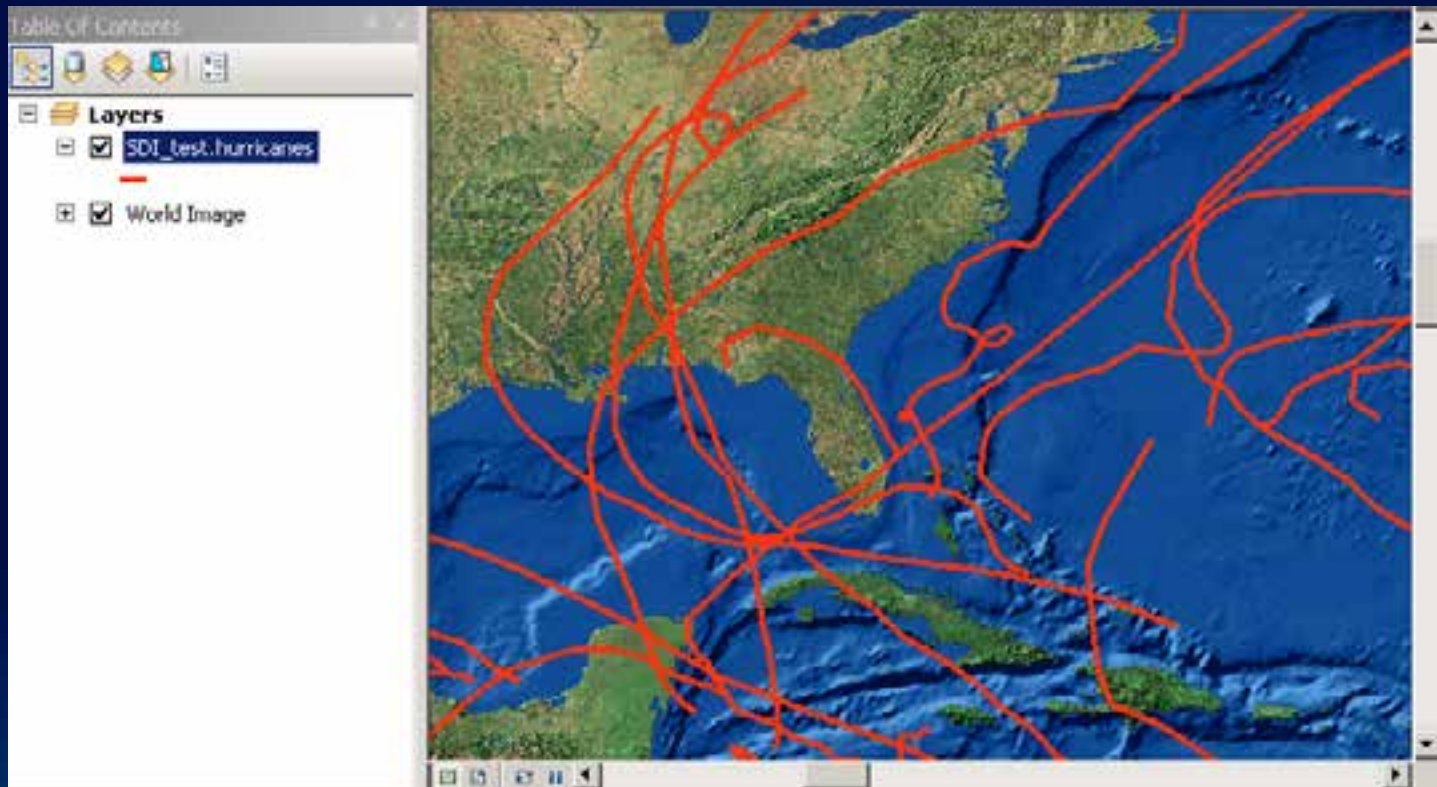
# Viewing database data in ArcGIS

- **Tables (with and without a spatial type) are viewed in ArcGIS through a query layer**
  - Define the layer yourself or let ArcGIS discover how to define it
- **Query Layer is a layer that is defined by a SQL query**
  - Provide data integration with geodatabases as well as from databases
  - Can quickly integrate spatial and nonspatial information into GIS projects independently of where and how that information is stored

# Viewing database data in ArcGIS

- Simple SQL query

```
SELECT * FROM dbo.HurricaneTracks_2005 hurricane
```



## Viewing database data in ArcGIS

- **Most complex SQL query that uses casting, derived columns and spatial operators**

```
SELECT county.id, county.State_name, county.NAME county_name,  
county.POP1990 population, CAST(county.POP1990 as  
decimal)/CAST(states.POP1990 as decimal)*100 PctStatePop,  
county.Shape FROM dbo.HurricaneTracks_2005 hurricane, dbo.counties  
county, dbo.states states WHERE hurricane.NAME = 'KATRINA' AND  
hurricane.Shape.STIntersects(county.shape) = 1
```

## Viewing database data in ArcGIS

- **Most complex SQL query that uses casting, derived columns and spatial operators**

```
SELECT county.id, county.State_name, county.NAME county_name,  
county.POP1990 population, CAST(county.POP1990 as  
decimal)/CAST(states.POP1990 as decimal)*100 PctStatePop,  
county.Shape FROM dbo.HurricaneTracks_2005 hurricane, dbo.counties  
county, dbo.states states WHERE hurricane.NAME = 'KATRINA' AND  
hurricane.Shape.STIntersects(county.shape) = 1
```

## Viewing database data in ArcGIS

- **Most complex SQL query that uses casting, derived columns and spatial operators**

```
SELECT county.id, county.State_name, county.NAME county_name,  
county.POP1990 population, CAST(county.POP1990 as  
decimal)/CAST(states.POP1990 as decimal)*100 PctStatePop,  
county.Shape FROM dbo.HurricaneTracks_2005 hurricane, dbo.counties  
county, dbo.states states WHERE hurricane.NAME = 'KATRINA' AND  
hurricane.Shape.STIntersects(county.shape) = 1
```

## Viewing database data in ArcGIS

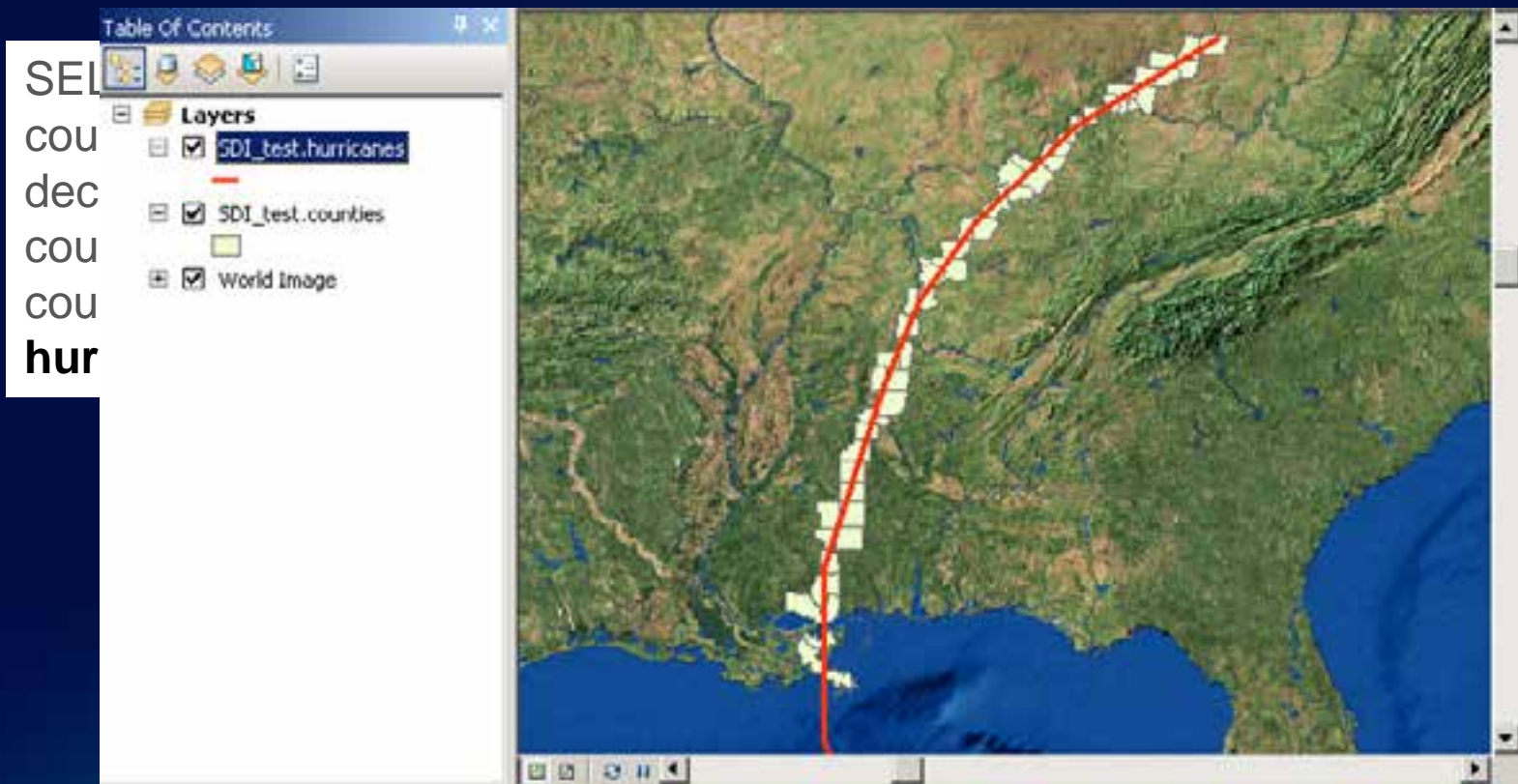
- **Most complex SQL query that uses casting, derived columns and spatial operators**

```
SELECT county.id, county.State_name, county.NAME county_name,  
county.POP1990 population, CAST(county.POP1990 as  
decimal)/CAST(states.POP1990 as decimal)*100 PctStatePop,  
county.Shape FROM dbo.HurricaneTracks_2005 hurricane, dbo.counties  
county, dbo.states states WHERE hurricane.NAME = 'KATRINA' AND  
hurricane.Shape.STIntersects(county.shape) = 1
```



# Viewing database data in ArcGIS

- Most complex SQL query that uses casting, derived columns and spatial operators



SEL  
cou  
dec  
cou  
cou  
hur

## Other Database Tasks

- **Connecting to a database**
- **Supported data types**
- **Viewing data and query layers**
- **Administer the database (e.g. grant access)**
- **Create new tables and alter schema**

# Building on top of Database Functionality

- **Display and use of the data is determined on the fly or defined by you within the layer**
- **Works well for viewing and querying spatial and non-spatial database information**
- **Cases where you want to do more with your data**

## **Building on top of Database Functionality...**

- **Store business rules with the data so they're available to everyone who accesses the data**
- **Advanced data modeling such as with transportation or facility networks**
- **Store and work with detailed cartography**
- **Multiple editors working on the same data at the same time without impacting each other**

# The Geodatabase

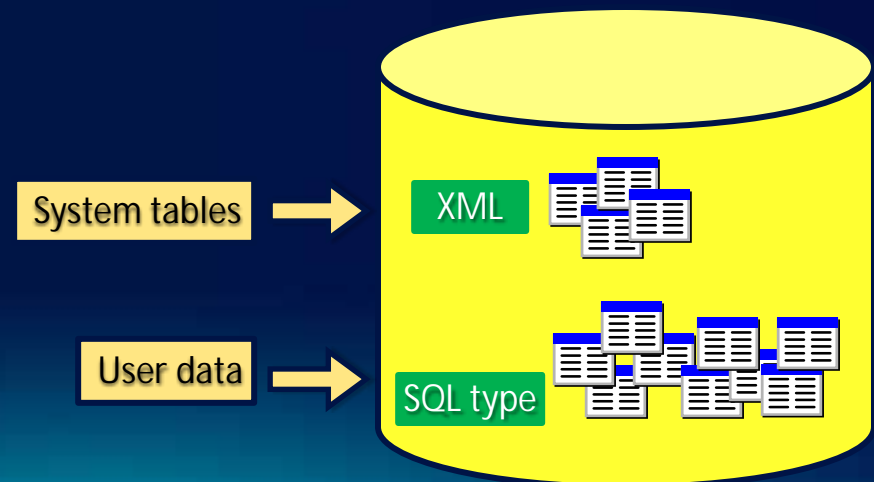
- **A physical store of geographic data**
  - Scalable storage model supported on different platforms
- **Core ArcGIS information model**
  - A comprehensive model for representing and managing GIS data
  - Implemented as a series of simple tables
- **A transactional model for managing GIS workflows**
- **Set of components for accessing data**

# **Geodatabase is based on relational principles**

- **The geodatabase is built on an extended relational database**
- **Leverages key DBMS principles and concepts to store geographic data as tables in a DBMS**
- **The core of the geodatabase is a standard relational database schema**
  - **a series of standard database tables, column types, indexes, and other database objects**

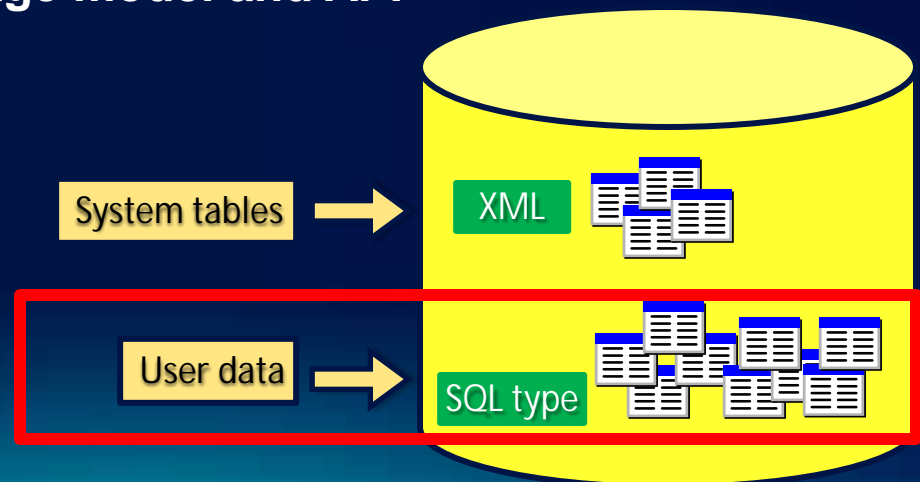
# Geodatabase Schema

- **There are two sets of tables:**
  - **Dataset tables (user-defined tables)**
  - **Geodatabase system tables**



# User-defined tables

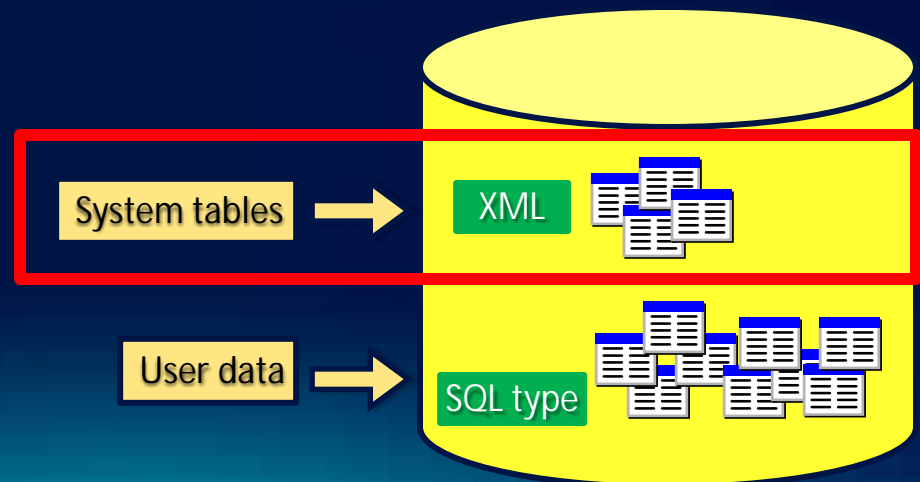
- Stores the content of each dataset in the geodatabase
- Datasets are stored in 1 or more tables
- Spatial Types enhance the capabilities of the geodatabase
  - SQL access to geometry
  - Industry standard storage model and API



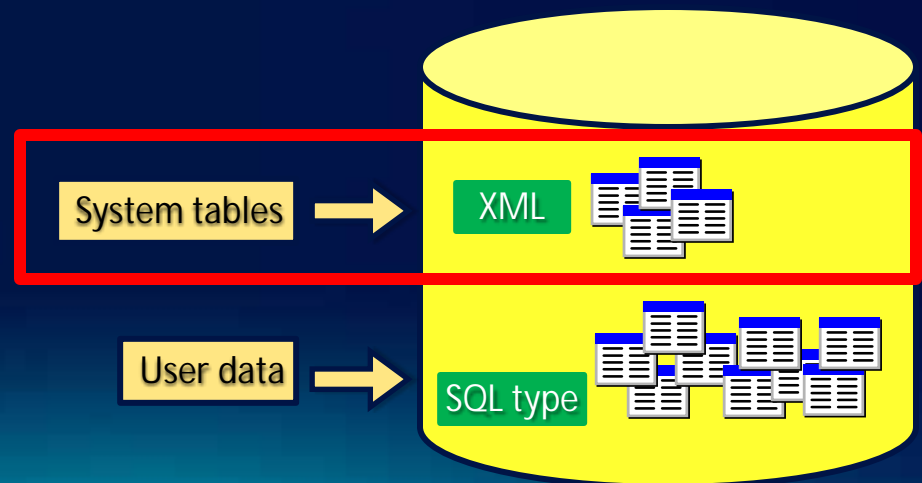
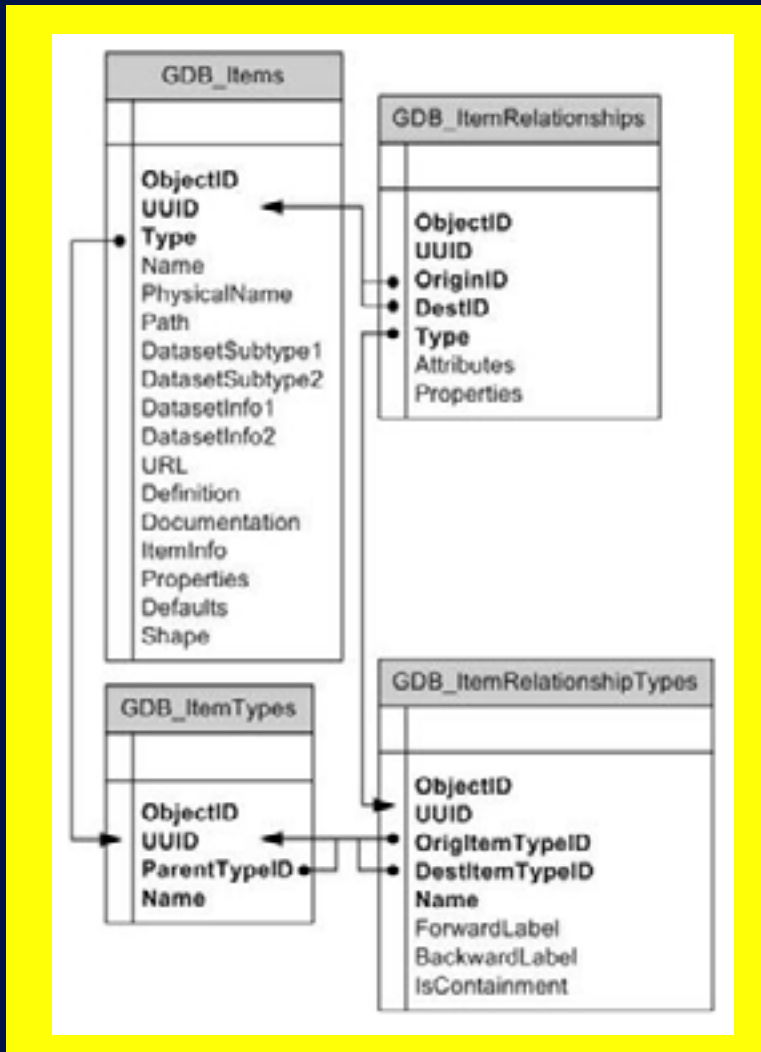


# Geodatabase system tables

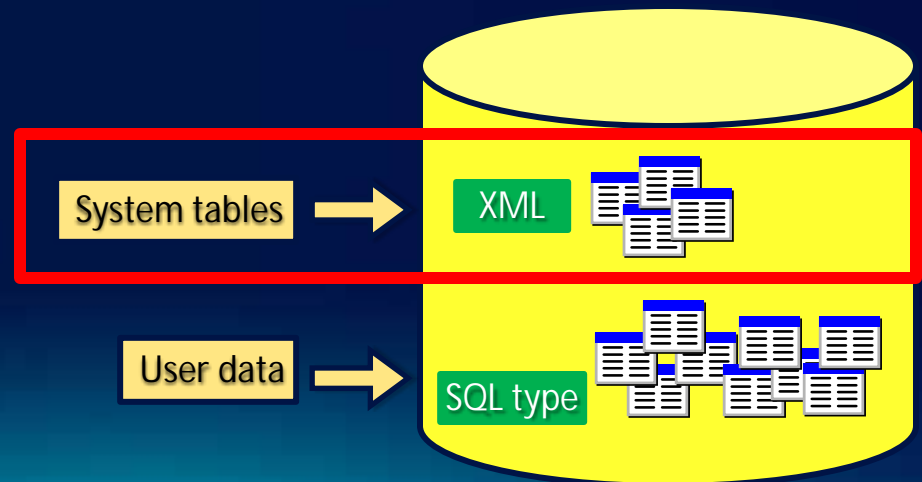
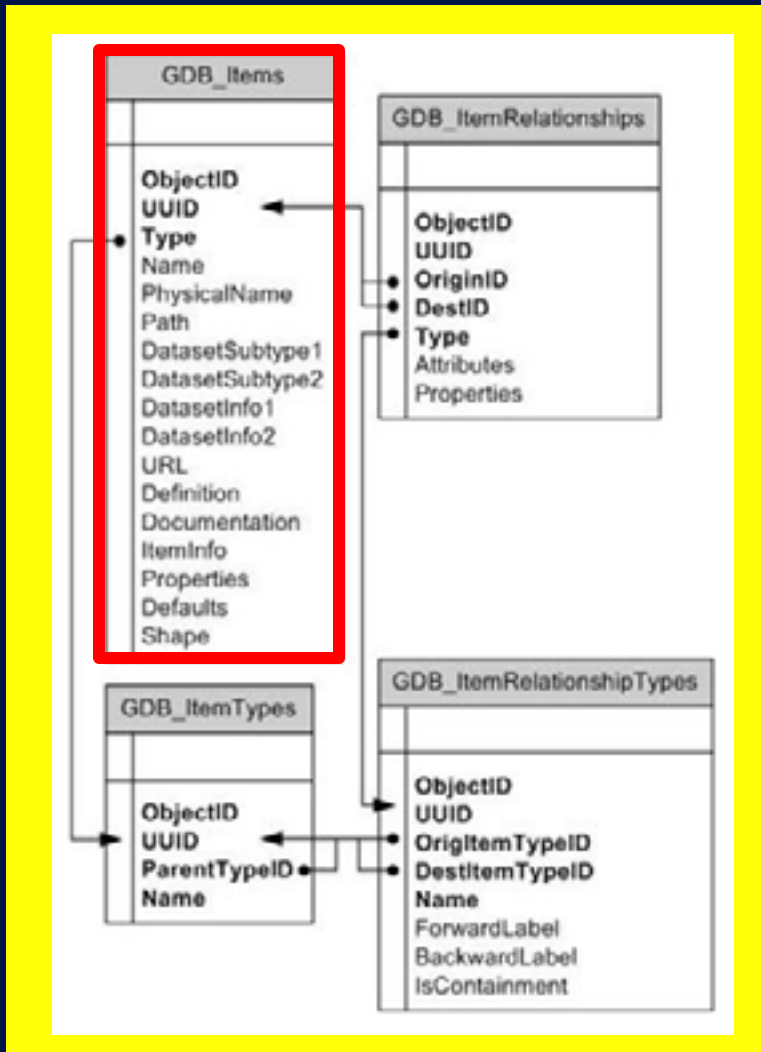
- **System tables store definitions, rules, and behavior for datasets**
- **Tracks contents within a geodatabase**
- **4 main system tables**
- **Geodatabase schema is stored primarily within an XML field**



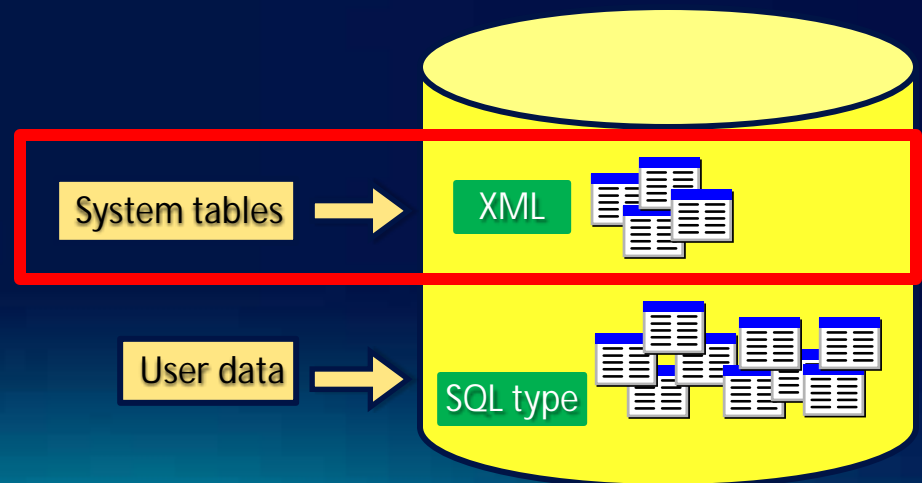
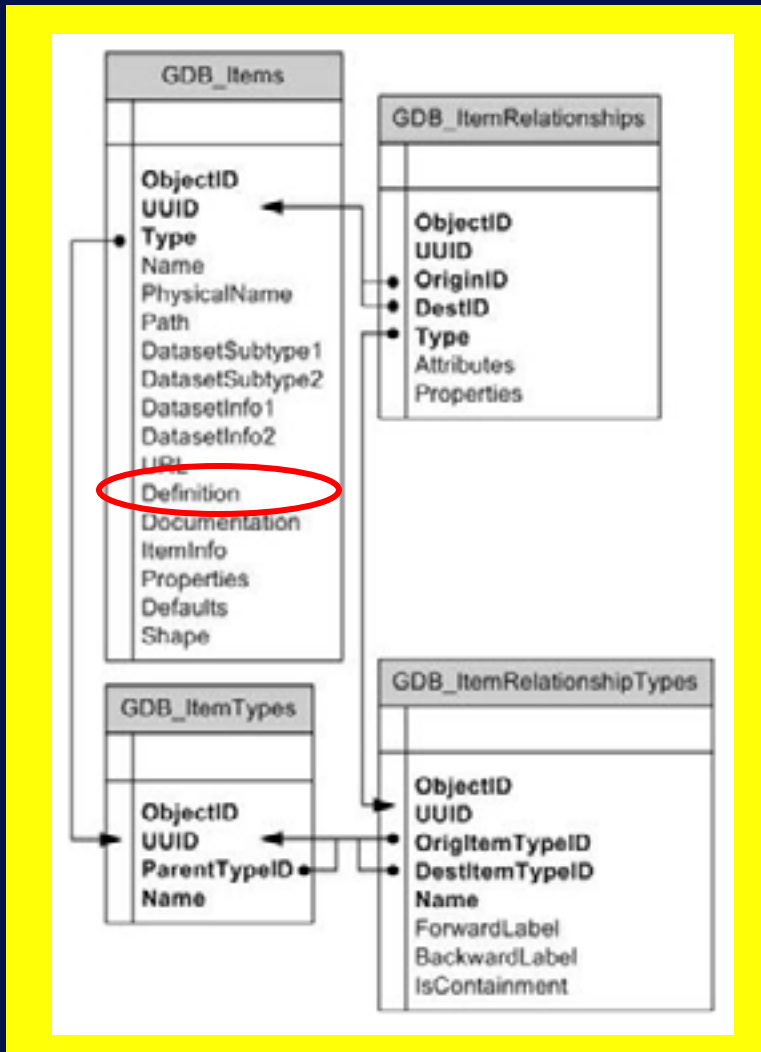
# Geodatabase Schema...



# Geodatabase Schema...



# Geodatabase Schema...



# Modeling Real-World Data with the Geodatabase ...

- The geodatabase enhances data and thematic layers by adding rules and behavior

- Spatial and relational integrity rules
- Data validation
- Business logic

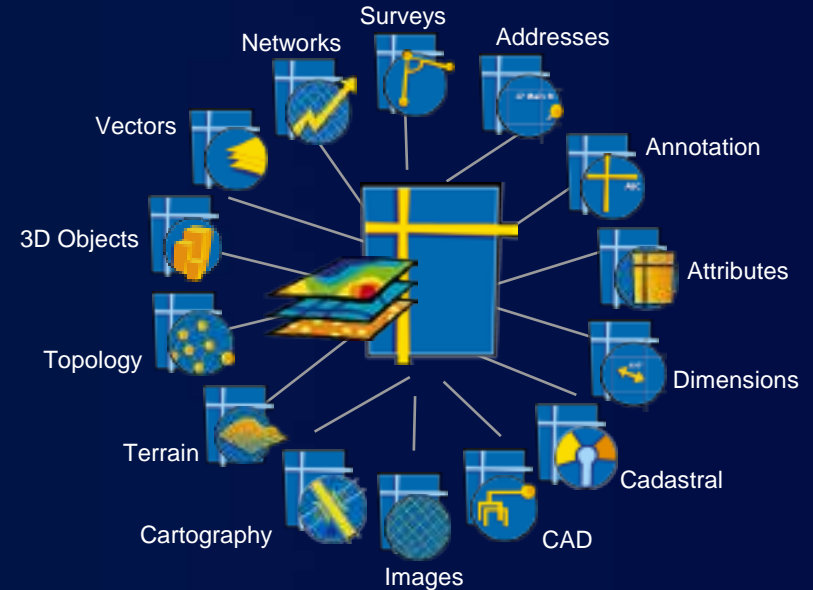
- Create thematic layers with behavior

- Road and utility networks
- Parcel fabrics
- Terrain and 3D surfaces
- Location services

- Extended framework for advanced workflows and editing

- Multiuser editing, Data Replication, Editor tracking, Archiving

## Geodatabase Functionality



# Presentation Outline

- Introduction
- Databases and Geodatabases
- **SQL**
- ArcObjects
- Plug In Data Sources
- Python
- File Geodatabase API
- Runtime
- Web Editing

# What is SQL (Structured Query Language)?

- **A set of defined expressions and syntax used to query and manipulate data in a DBMS**
- **Most SQL based on an ANSI standard and then extended for each DBMS**
  - **SQL syntax across different DBMSs is slightly different**
- **SQL can be used to access, create, and update data**
- **Geometry is stored within a Spatial Type**

# What is a spatial type?

- **A spatial type (ST) is a type that stores geometry data in a single spatial attribute**
  - **Geometry type, coordinates, dimension, spatial reference**
- **Spatial Index**
  - **Access path for quick retrieval**
- **Relational and geometry operators and Functions**
  - **Constructors**
  - **Accessor**
  - **Relational**
  - **Geometry**



# What are the benefits of a spatial type?

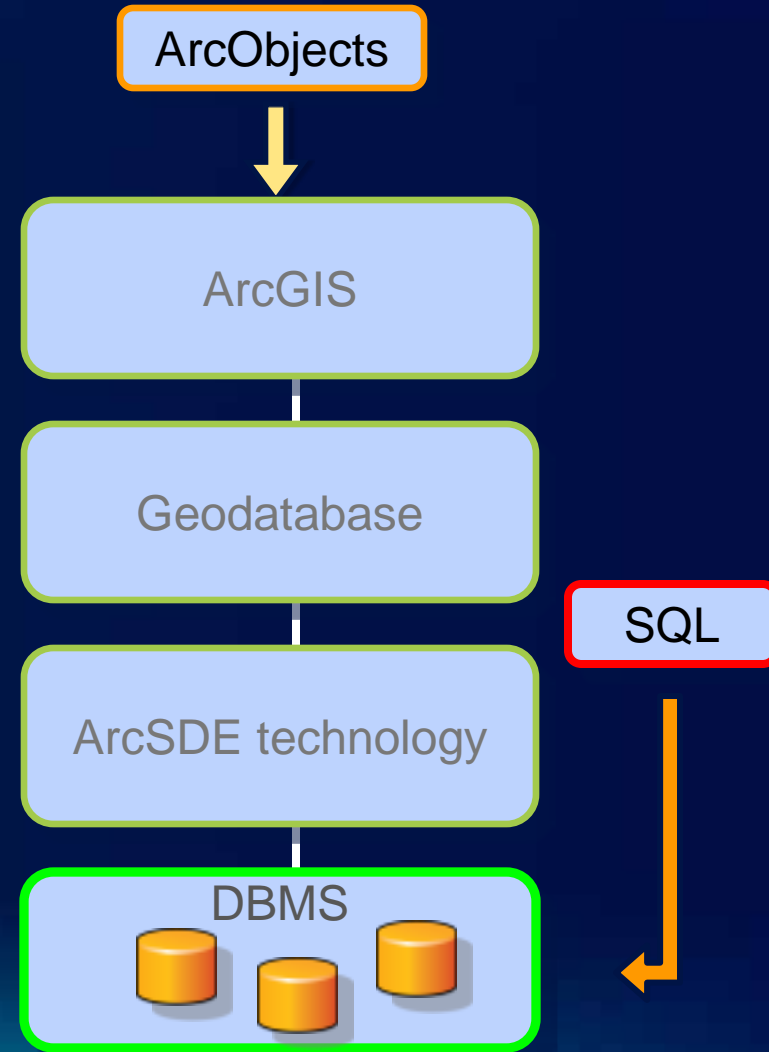
- **Efficiency**
  - Spatial data and methods are stored in the database
  - Applications access native dbms type
- **Accessed using common API's and SQL**
  - C, C++, C#, Java, OLEDB
  - Adheres to standards for SQL access
- **With SQL and a ST you can**
  - Create tables with a spatial attribute
  - Read and analyze the spatial data
  - Insert, update, and delete simple geometry data

# Accessing Geodatabase through SQL

- **Access schema and properties of existing datasets**
  - Use SQL SELECT statements to query the Definition field in the GDB\_ITEMS table
- **Editing tables/feature classes, versioned or not**
- **Create tables with or without spatial type**
- **Use spatial SQL functions to evaluate attributes and spatial relationships, perform spatial operations, and return and set spatial properties**

# Accessing Geodatabase through SQL

- With SQL accessing the data at the DBMS level
  - Bypass behaviors and functionality enforced by the geodatabase or ArcGIS clients
- Need to be aware of what you can and cannot edit
  - Relationship classes
  - Geometric networks
  - Topology...



# Accessing Geodatabase through SQL

- In the context of the Geodatabase, creation and update limited to simple data
  - Attributes of user-defined tables or feature classes in an enterprise geodatabase if those attributes do not participate in geodatabase functionality
  - User-defined database tables that are not registered with the geodatabase
- Use the **Is\_Simple** function to determine whether your data can be updated

# Schema Creation

- **You can use the native SQL of your DBMS to tables and populate tables**
  - **Need to register the table with the geodatabase to use geodatabase functionality**
- **Generally advise to not modify schema of geodatabase items (i.e domains) through SQL**

```
CREATE TABLE hazardous_sites (row_id integer NOT NULL, site_id integer, name varchar(40), location sde.st_geometry);
```

Register with Geodatabase command

# Editing tables/feature classes

- **Editing ArcGIS feature classes with SQL**
  - Simple features only
  - Points, lines, polygons (single or multipart)
  - Ability to modify geometry when stored as a spatial type
  - Without geodatabase behavior
    - Not part of topology, geometric network, etc...
- **Editing tables/feature classes**
  - Use SQL SELECT statements
  - Directly editing the database tables (no delta tables)
  - Nonversioned editing in ArcGIS terminology

# Editing tables/feature classes

- **Editing versioned tables/feature classes**
  - Requires multiversioned views
- **Can use SQL to update, insert and delete data from tables that are not versioned**
- **Can leverage DBMS functionality**
  - Unique indexes, constraints, referential integrity, default values, triggers

## Editing tables/feature classes

- **Requires a unique identifier (ObjectID) when inserting**
  - **Used to uniquely identify rows in tables in a geodatabase**
  - **Obtained from classes sequence or procedure**
  - **Object ID is used by ArcGIS to do such things as display selection sets and perform identify operations on features**



## Editing versioned tables/feature classes

- **Must use several stored procedures/commands installed with ArcSDE technology**
  - Create multiversioned views (sdetable –o create\_mv\_view)
  - Create a new version (create\_version)
  - Set which version to access (set\_current\_version)
  - Perform edits within the new version (edit\_version )
- **Unlike nonversioned editing, ObjectID values for new records are automatically generated**
  - Changes are made to the delta tables
  - Versions must be reconciled through ArcGIS

# Accessing Geodatabase through SQL

- Geodatabase built on simple tables and well-defined types
- SQL can be used to access, create, and update simple data
  - Access schema and properties of existing datasets
  - Editing tables/feature classes even when versioned
  - Spatial and attribute queries
  - Schema operations limited to table creation
- Query Layers provide way to integrate results of SQL operations into ArcGIS

# Demo

- SQL access with the Geodatabase demo

# Presentation Outline

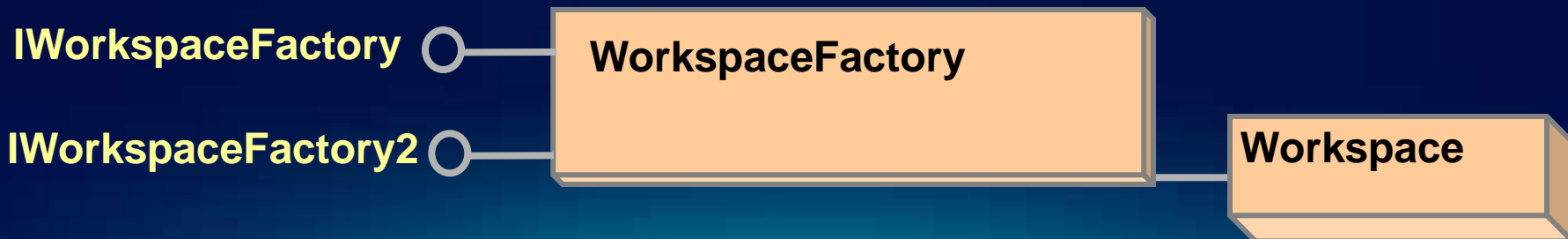
- Introduction
- Databases and Geodatabases
- SQL
- **ArcObjects**
- Plug In Data Sources
- Python
- File Geodatabase API
- Runtime
- Web Editing

# What is ArcObjects?

- **ArcObjects is a library of Component Object Model (COM) components that make up the foundation of ArcGIS**
- **ArcObjects components are installed with the ArcGIS Desktop, ArcGIS Engine, or ArcGIS Server products and can be used in a number of ways:**
  - **To customize the ArcGIS Desktop applications**
  - **To build standalone mapping applications**
  - **To develop Web applications**
- **Grouped into over 65 assemblies based on relationships between objects**
- **Available for .Net, Java, C++**

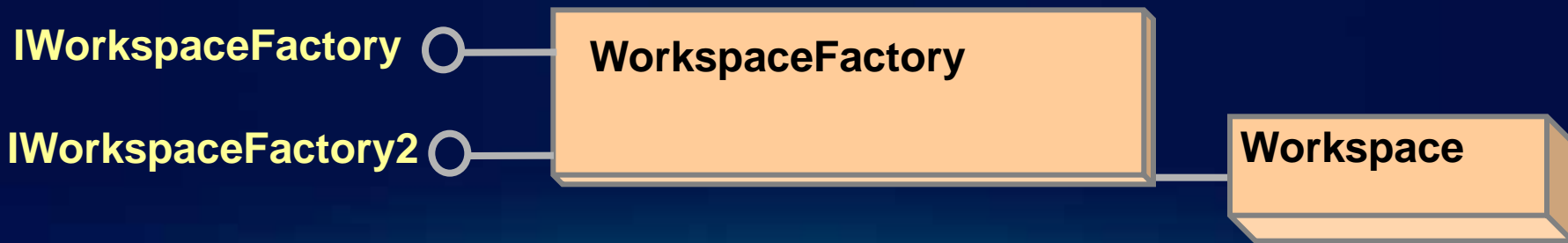
# Workspace Factory and Workspaces

- **Workspace Factory is a dispenser of workspaces**
  - Personal, File, ArcSDE workspaces, **SQL workspaces**  
Create a factory from a workspace factory coclass
- **Workspace factories are singleton objects**
  - One instance created per Component Object Model (COM) apartment
  - Further calls return a reference to the existing object



# Workspace Factory and Workspaces

- A workspace is a container of datasets.
  - Database or Geodatabase
- Workspaces are not singleton objects
  - But, requesting a workspace with the same properties as an existing instance returns a reference to it
  - The Geodatabase guarantees unique instancing



# Making a connection

- Create a workspace from a factory
  - Path to data and window handle (app ID)

```
IWorkspaceFactory sdeWkspFact = new SdeWorkspaceFactoryClass();
IPropertySet propset = new PropertySetClass();

propset.SetProperty("SERVER", "crimsontide");
propset.SetProperty("INSTANCE", "5151");
propset.SetProperty("USER", "brent");
propset.SetProperty("PASSWORD", "brent");
propset.SetProperty("DATABASE", "null");
propset.SetProperty("VERSION", "SDE.DEFAULT");
propset.SetProperty("AUTHENTICATION_MODE", "DBMS");

IWorkspace workspace = sdeWkspFact.Open(propset, 0);
```

- Connecting using a connection file

```
string nameOfFile = "D:\\data\\redarrow.sde";
IWorkspace workspace = workspaceFactory.OpenFromFile(nameOfFile, 0);
```



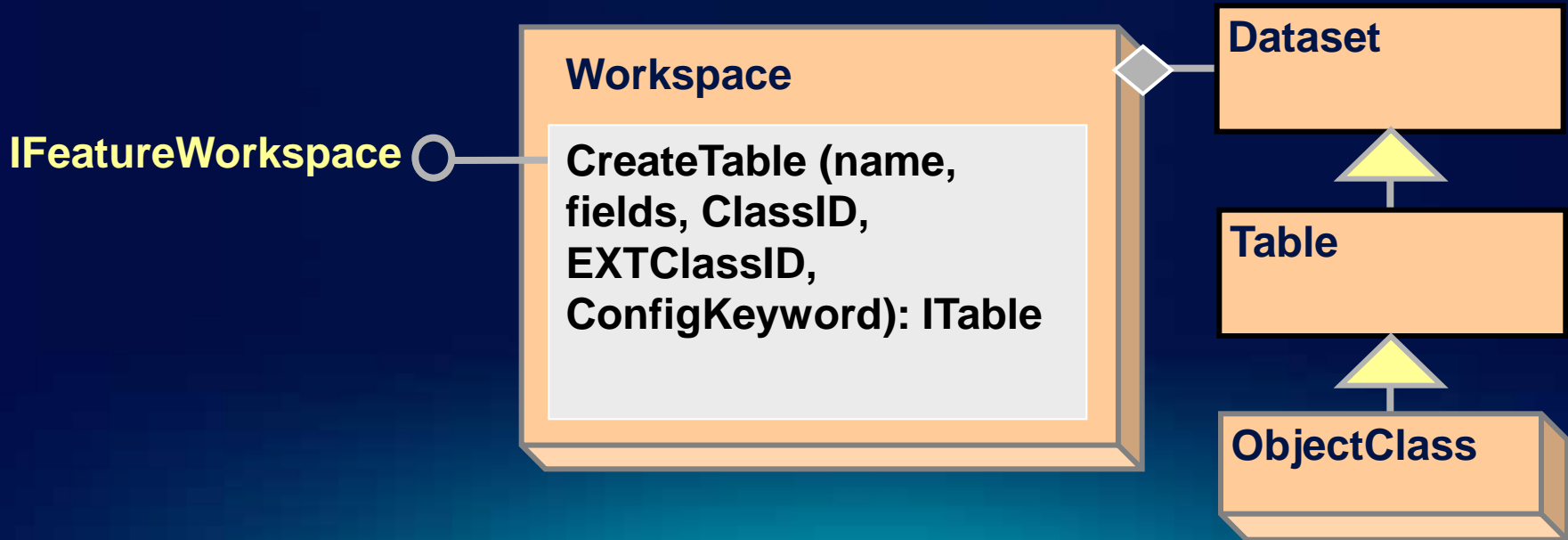
# Demo

---

- **Database Access with ArcObjects**

# Create an Object Class

- Creates an ObjectClass, returns ITable interface
  - ObjectID field. Values are Never Reused.
  - From a database perspective, can be viewed as a primary key
- Can also use it to create a Table
- Custom behavior ID's (*can use null for EXTClassID*)
- Configuration keywords (*can use " "*)
- Can create Fields first : IObjectClassDescription:RequiredFields

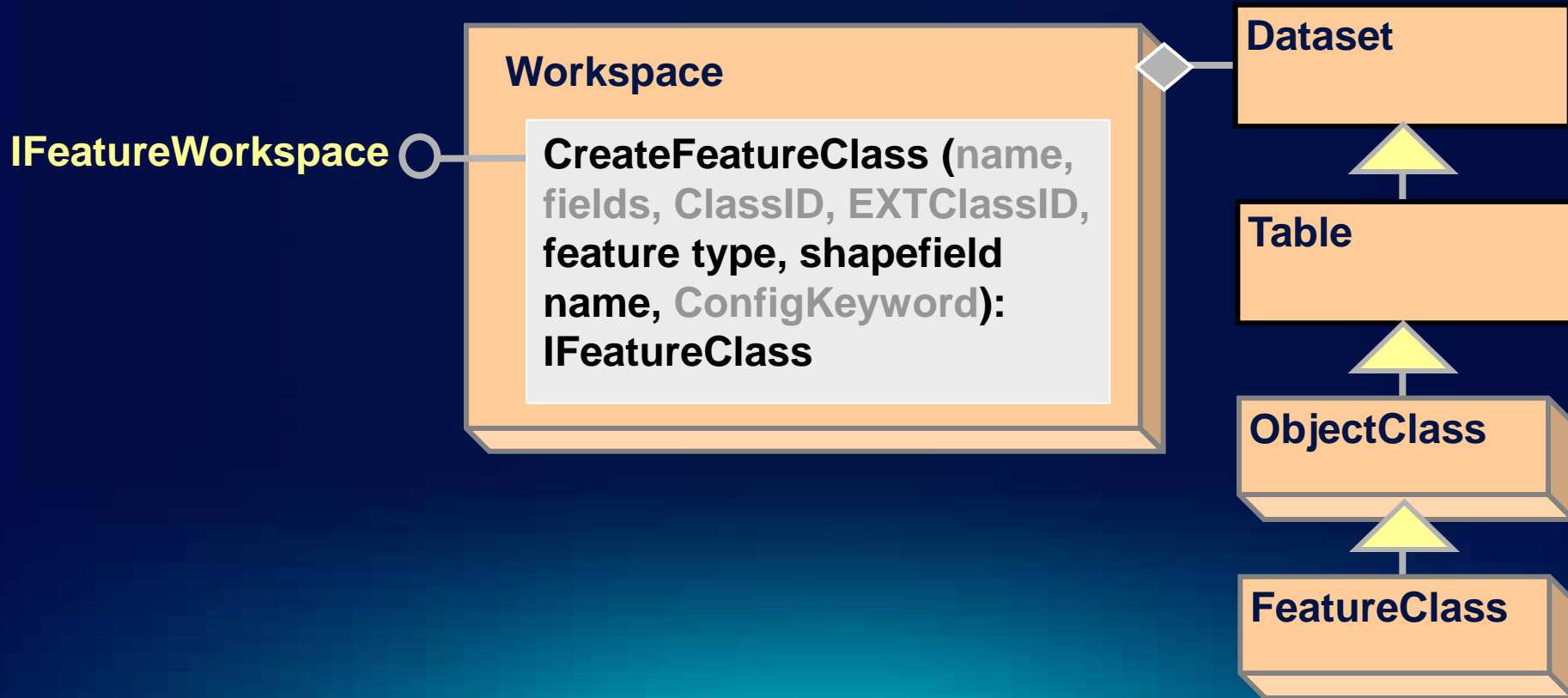


## Create an Object Class ...

- **Object Classes** have geodatabase specific functionality that Tables do not, through several interfaces
  - **IObjectClass**
    - **AliasName**
    - **ObjectClassID**
    - **RelationshipClasses**
  - **ISubtypes**
    - **AddSubtype**
- **Standard table operations**, such as queries and row creation, are still performed on an object class using the **ITable** interface.

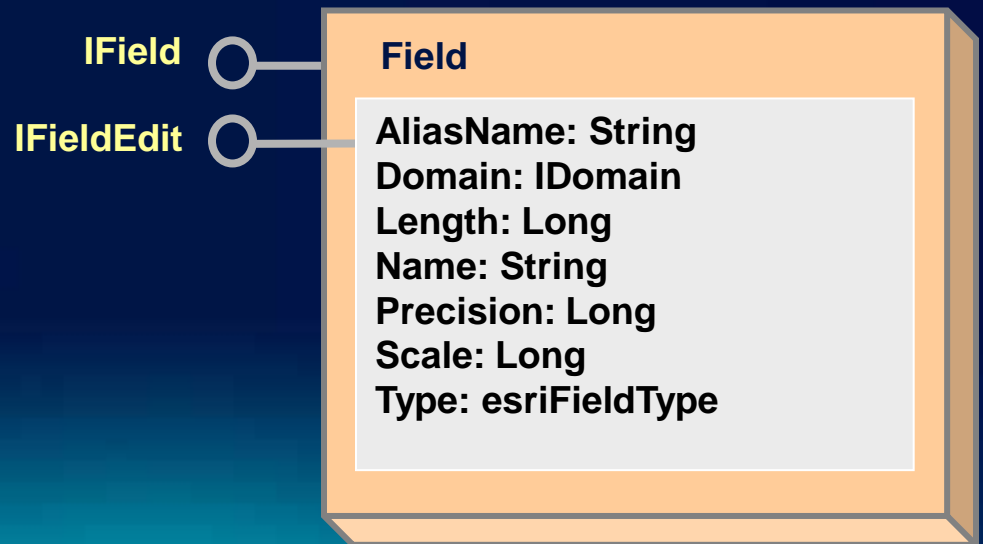
# Create a Feature Class

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



# Create a Field

- Synonymous with a column
- Set properties with IFieldEdit
- Geodatabase tables/feature classes must have an ObjectID field
  - Using Class Descriptions will take care of this
- Use a Field collection (Fields) when creating datasets
  - For existing tables use IClass::AddField method to add fields and IClass::DeleteField method to delete fields
- Set properties for the Field with the IFieldEdit interface



# Feature Datasets ...

- A container object for datasets with the same spatial reference
  - Only exist within a Geodatabase
- Need to remember that feature classes may or may not belong to a feature dataset
  - The following code assumes a feature dataset exists and may fail:

```
IFeatureDataset featureDataset = featureClass.FeatureDataset;  
IWorkspace workspace = featureDataset.Workspace;
```

- This piece of code will work for standalone feature classes and those in feature datasets:

```
IDataset dataset = (IDataset)featureClass;  
IWorkspace workspace = dataset.Workspace;
```

# Domains

- Describe the legal values of a field type.
  - Used to ensure attribute integrity
- Use the **IWorkspaceDomains** interface to manage the collection of domains found within a workspace.
  - Since Domains are shared amongst feature classes; the management of them is at the workspace level
- **DeleteDomain** will fail if the domain is associated with a field
- Domain names are unique across a workspace
  - Need to check for existence of Domain name prior to creation or trap for the error

**IWorkspaceDomains** ○

## Workspace

AddDomain  
AlterDomain  
DeleteDomain  
DomainsByFieldType  
DomainsByName  
...

# Subtypes

- **Subtypes are specific to feature class**
  - Partition the objects in a class into like groups
- **Once set, need to check attribute, connectivity and relationship rules at subtype level**
- **Each object class has a default subtype code**
  - Important for feature creation and editing
- **ISubtypes interface is used for managing subtypes and the associated default values and attribute domains**

**ISubtypes** ○

**ObjectClass**

**AddSubtype**  
**DefaultSubtypeCode**  
**Domain**  
...



# Creating 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 record in the table

# Simple vs. Complex Features

- **Within the geodatabase, behavior is dependent upon whether a feature is simple or complex**
- **Simple features**
  - Point, line, polygon, multipoint, multipatch features
  - Simple Relationships
- **Complex features**
  - Network features (simple edge, simple junction, complex edge)
  - Annotation features
  - Dimension features
- **Any dataset specific behavior (i.e.: for features created in geometric networks) is handled at creation time**
  - Not required to call Connect or create Dirty Areas

# Demo

---

- **Creating Features**
  - Walk through basic process to create a feature

# Geodatabase Editing - Edit Session

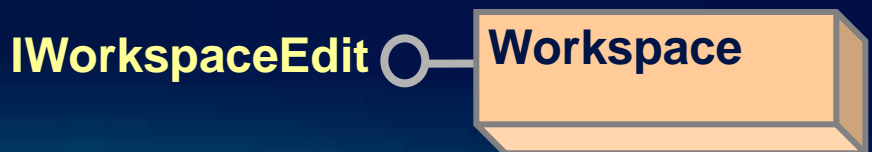
- **Geodatabase explicitly stores change information when edited**
- **Only see the changes you've made within the edit session**
  - Changes made by other applications are not seen
  - Until Save or Discard
- **Edits should be made within an edit operation**
  - StartEditOperation – StopEditOperation
  - Perform the edit as quickly as possible
  - Keep edit operation “tight and compact”
  - Collect the required information before starting the edit operation

## Geodatabase Editing - Edit Session ...

- **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**
    - **Pushes the edit operation onto the undo stack**
- **UndoEditOperation, RedoEditOperation**
  - **Geodatabase moves the operation between the undo and redo stacks**

# Editing the Geodatabase

- **When to use edit sessions?**
  - Must use with topologies, geometric networks, etc
  - 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**



## Other Useful Method When Editing

- **IDatasetEdit.IsBeingEdited**
  - Determine if a particular dataset is participating in the edit session
- **IWorkspaceEdit2.IsInEditOperation**
  - Determine if the workspace is currently in an edit operation
  - Use when deciding whether to start an edit operation
- **IWorkspaceEdit2.EditDataChanges**
  - Determine which features have been changed with the scope of an edit session or edit operation.

# Editing Demo

- Edit Operations

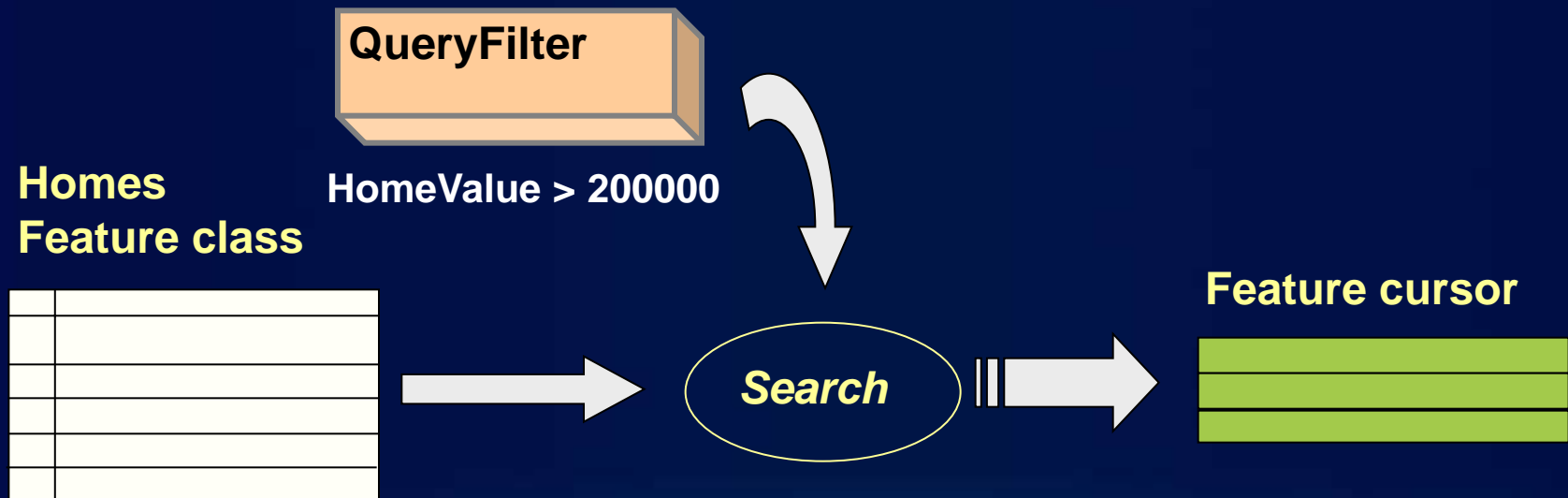


# Cursors

- **A geodatabase object used for the iteration of records returned from a query**
- **3 Class Cursors**
  - Search (general query cursor)
  - Update (positioned update cursor)
  - Insert (bulk inserts)
- **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 a SQL-like statement
- The cursor contains a subset
  - No filter\nnothing, all rows returned



# IQueryFilter

```
IQueryFilter queryFilter = new QueryFilterClass();
queryFilter.SubFields = "OBJECTID,FULLNAME,ParcelID";
queryFilter.WhereClause = "FULLNAME like 'D%'";

IQueryFilterDefinition queryFilterDef =
(IQueryFilterDefinition)queryFilter;
queryFilterDef.PostFixClause = "ORDER BY FULLNAME";

IFeatureCursor featureCursor = featureClass.Search(queryFilter, true);
```

## 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**

# ISpatialFilter

- Used to query spatial aspects of a feature class
  - Inherits from IQueryFilter

```
ISpatialFilter spatialFilter = new spatialFilterClass();  
  
spatialFilter.SubFields = "OBJECTID,FULLNAME,ParcelID,SHAPE";  
spatialFilter.Geometry = envelope;  
spatialFilter.SpatialRel = within;  
spatialFilter.WhereClause = "FULLNAME like 'D%'";  
  
IFeatureCursor featureCursor = featureClass.Search(spatialFilter, true);
```

# Types of Class Cursors

- Search cursors
  - Returns rows specified by a Query or Spatial Filter
- Update cursors
  - Update and delete rows specified by the filter
  - Specify the ObjectID field
- Insert cursors
  - Used for inserting rows into a table
- Accessed by
  - Corresponding methods on table or feature class

## Types of Class Cursors ...

- Forward only, do not support
  - Backing up and retrieving rows already retrieved
  - Making multiple passes
  - Resetting
- Solution:
  - Re-execute the query
- Release Class Cursors with
  - `Marshal.ReleaseComObject`
  - `Cleaner.release()`

# Types of Class Cursors ...

- **Insert cursors are used to bulk insert rows**
  - **Faster for loading simple data than IFeature.Store**
    - **Bypasses 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 database 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**



## Types of Class Cursors ...

- **Scope cursors to edit operations**
- **Cursor is bound to a specific state of the geodatabase**
- **When state of the geodatabase changes cursor is no longer valid and should not be used**
  - **Performing edits on a cursor that is incorrectly scoped can cause unexpected behavior.**

# Recycling Method

- **Recycling**
  - A recycling cursor is a cursor that does not create a new client side row object for each row retrieved from the database
  - Allocate a single row object
    - Re-hydrate on each fetch
  - Performance advantages
  - Primarily used for reading data
- **Non Recycling**
  - A different row object on each fetch
  - Always has full set of fields, even if `IQueryFilter::Subfields` used

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

# Cursor demo

---

- **Cursors examples**
  - **Search**
  - **Update**
  - **Insert**

# Presentation Outline

- Introduction
- Databases and Geodatabases
- SQL
- ArcObjects
- **Plug In Data Sources**
- Python
- File Geodatabase API
- Runtime
- Web Mapping

# What are Plug-In Data Sources?

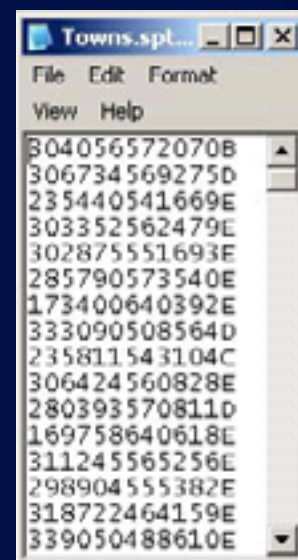
- **Integrates a new data format into ArcGIS in a read-only manner**
- **Provides ability to:**
  - **browse, preview, and manage the data in Catalog**
  - **select, render, query, label, and join the data in ArcMap.**
  - **program with the data source using normal geodatabase interfaces such as IWorkspace and IFeatureClass.**
- **Supports simple feature types; tables, feature classes and feature datasets**

## The Case for a Plug-In Data Source

- **Receive a regular supply of text files containing geographic locations**
  - **Data in the files has an unusual/non-standard format**
- **Don't want to convert data every time a new file is received to use with ArcGIS**
  - **Need to work with data in its native format within ArcGIS**

# The Case for a Plug-In Data Source

- **Example ASCII text file**
  - **Point data**
- **The first six characters are the x-coordinate**
- **The next six characters contain the y-coordinate**
- **The trailing characters contain an attribute value.**



A screenshot of a text editor window titled "Towns.spt...". The window contains a list of ASCII strings representing point data. Each string consists of 12 characters: the first six are the x-coordinate, the next six are the y-coordinate, and the last one or two characters are the attribute value. The data is as follows:

ASCII String
304056572070B
306734569275D
235440541669E
303352562479E
302875551693E
285790573540E
173400640392E
333090508564D
235811543104C
306424560828E
280393570811D
169758640618E
311245565256E
298904555382E
318722464159E
339050488610E

# Creating a Plug-In Data Source

- **To make a plug-in data source, you must implement four required classes:**
  - **plug-in workspace factory helper**
  - **plug-in workspace helper**
  - **plug-in dataset helper**
  - **plug-in cursor helper**
- **Other optional classes that can also be implemented**



# Demo

---

- **Creating a Plug-In Data Source**

# Presentation Outline

- Introduction
- Databases and Geodatabases
- SQL
- ArcObjects
- Plug In Data Sources
- **Python**
- File Geodatabase API
- Runtime
- Web Mapping

# What is Python?

- **Cross-platform, open-source programming language**
- **Does not require a compiler**
- **Programs that you write with Python are called scripts.**

# Why Python?

- Free
- Simple and easy to learn
- Modular
- Object oriented
- Easy to maintain
- Scalable
- Cross platform (Windows & UNIX/Linux)
- Wide-acceptance



## Why Python ...

- **Significantly reduces the amount of time spent on a project.**
  - Quickly execute & re-execute tools or functions
  - Automate common tasks
- **It is designed to be easy to read and learn**
  - **“Maintainability”** – easy to modify and keep up to date

# What is ArcPy?

- **A cornerstone for automation in ArcGIS**
  - Data analysis, data conversion, data management, map automation, **geodatabase management**
- **ArcPy is a native Python site-package**
  - Site package is Python's term for a library that adds additional functions to Python
  - Access to 800+ geoprocessing tools
  - Embedded reference documentation for each function, class and module
  - Code completion for ArcGIS components in your favorite Python editor

# Running Python Code

- **Through the Python window**
  - An interactive window that lets you enter Python code, execute it, and view the results in your active map
- **Script Tool**
  - Create a geoprocessing tool that executes a Python script
- **Operating system prompt**
  - Execute a Python script that uses ArcPy routines
  - Need an ArcGIS application, does not need to be running

## **New Data Access module at 10.1**

- **Improved cursor support (faster performance)**
- **Control of the edit session, edit operation**
- **Functions for converting tables and feature classes to and from NumPy arrays**
- **Support for versioning, replicas, domains, and subtypes workflows**



# Demo

- Python demo

# Presentation Outline

- Introduction
- Databases and Geodatabases
- SQL
- ArcObjects
- Plug In Data Sources
- Python
- **File Geodatabase API**
- Runtime
- Web Editing

# File Geodatabase API

- **Provide a non-ArcObjects based means by which advanced developers can work with File Geodatabases**
- **C++ API with coarse grained access to File Geodatabase**
- **Will not replace other APIs as the recommended approach to interacting with the File Geodatabase**



## **File Geodatabase API...**

- **Leveraging the work done with simplifying the Geodatabase**
  - Will only support file geodatabases created with 10.0 and newer clients
  - No support for pre-10.0 file geodatabases
- **Target audience**
  - Advanced developers who require access to the File Geodatabase without an ArcObjects license for purposes of interoperability

# Coarse-Grained Tasks possible with API

- **Create, Open, Delete file geodatabases**
- **Read the schema of a geodatabase**
  - **All content within a geodatabase can be opened for read access**
- **Create schema for objects within the simple feature model:**
  - **Tables**
  - **Point, Line, Polygon feature classes**
  - **Feature datasets**
  - **Domains**
  - **Subtypes**

## Coarse-Grained Tasks possible with API...

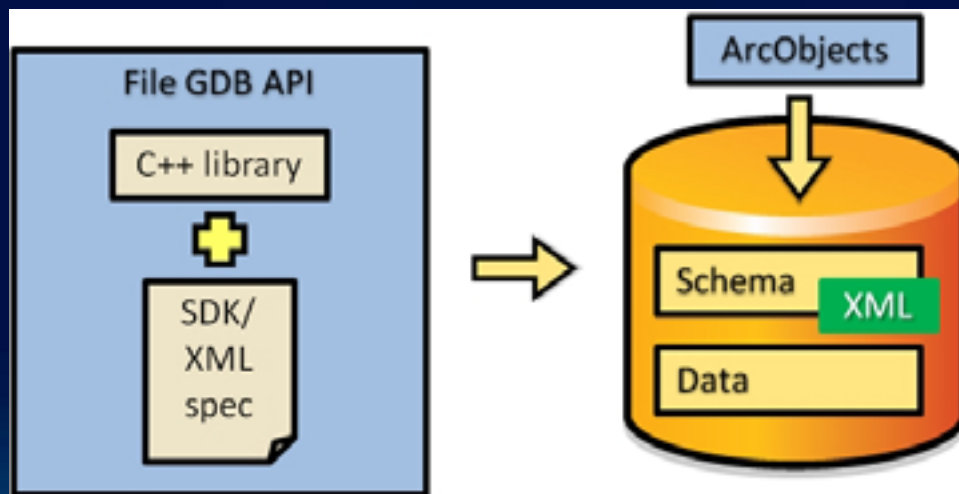
- **Read the contents of datasets in a geodatabase**
  - The majority of dataset content within a geodatabase can be read
    - Some exceptions such as network indexes
- **Insert, Delete and Edit the contents of simple datasets:**
  - Tables
  - Point, Line, Polygon, Multipoint, Multipatch feature classes

## Coarse-Grained Tasks possible with API...

- **Perform attribute and (limited) spatial queries on datasets**
  - Spatial queries will be limited to the envelope-intersects operator
- **Spatial References are limited to pre-defined GCS, PCS and Unknown**
  - Custom coordinate systems are not supported
- **Supports for a subset of the SQL 92 standard**
  - e.g. Select statements, Order By, Joins

# File Geodatabase API Overview

- Single downloadable ZIP file containing:
  - C++ library (single dll, lib, .h) built on Windows and Linux platforms
  - API documentation (html) and Samples
- Freely available from the [Geodatabase Resource Center](#)





# Demo

- **File API demo**

# Presentation Outline

- Introduction
- Databases and Geodatabases
- SQL
- ArcObjects
- Plug In Data Sources
- Python
- File Geodatabase API
- **Runtime**
- Web Editing

# Lightweight Applications and Runtime SDKs

Configurable applications and native ArcGIS Runtime SDKs

iOS	Android	Windows Phone	Windows Mobile	Windows	Linux
<b>Applications</b> <ul style="list-style-type: none"><li>• ArcGIS</li></ul>	<b>Applications</b> <ul style="list-style-type: none"><li>• ArcGIS</li></ul>	<b>Applications</b> <ul style="list-style-type: none"><li>• ArcGIS</li></ul>	<b>Applications</b> <ul style="list-style-type: none"><li>• ArcGIS</li></ul>	<b>Applications</b> <ul style="list-style-type: none"><li>• ArcGIS Explorer*</li></ul>	<b>Applications</b> <ul style="list-style-type: none"><li>• None</li></ul>
<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• Objective C</li></ul>	<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• Java</li></ul>	<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• Silverlight</li></ul>	<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• .NET</li></ul>	<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• WPF, Java, Qt</li></ul>	<b>Runtime SDK</b> <ul style="list-style-type: none"><li>• Qt, Java</li></ul>
					

# Introducing the ArcGIS Runtime for Desktop

## A GIS Runtime for Windows and Linux

- **Integrated into the ArcGIS Ecosystem**
- **Small Footprint**
- **Fast Display**
- **Easy to Deploy**

*Leverages the ArcGIS System*

# ArcGIS Runtime for Desktop Specifics

- **New Architecture**
  - Native 32 and 64 bit code execution
  - Utilizes hardware (Cores, CPUS,..)
  - Asynchronous programming pattern
- **Simplified Deployment**
  - No install required
  - Deploy only needed components
  - Side-by-Side deployment
  - Independent of other ArcGIS installs
- **SDKs**
  - WPF, Java, Qt,
  - Builds on the ArcGIS APIs

# Its Not Just About Writing Code

- **ArcGIS Desktop Used to Provision Content**
- **Content is Authored**
  - **Map Packages**
  - **Tile Packages**
  - **Locator Packages**
- **Functionality Can Be Authored**
  - **Geoprocessing Packages**
- **Packages can be delivered on Media, or downloaded from online**
- **ArcGIS Server Can Deliver Content to Clients**



## Available Functionality

- **Fully Supports ArcGIS Cartography Including Representations, Annotation and Labeling**
- **Geocoding**
- **Geoprocessing Tools, Scripts and Models Supported**
  - **System Tools Available Depend on Level of the Runtime**
- **Geodatabase Feature Editing**
  - **File and Enterprise Geodatabase**
  - **Simple Feature Editing**
  - **Attributes and Shape**
- **Feature Services**



## Desktop Developer

# Development Options





# Demo

- Runtime demo

# Presentation Outline

- Introduction
- Databases and Geodatabases
- ArcObjects
- SQL
- Python
- File Geodatabase API
- Runtime
- **Web Editing**

# Agenda

- **The basics of Web Editing in ArcGIS Server**
- **Web Editing scenarios**
  - Attachments
  - Editor tracking
  - Ownership Based Access Control
  - Handling complex symbology/geometries
  - Disable geometry edits
  - Editing versions
  - Sophisticated Web Editing
- **Q&A**

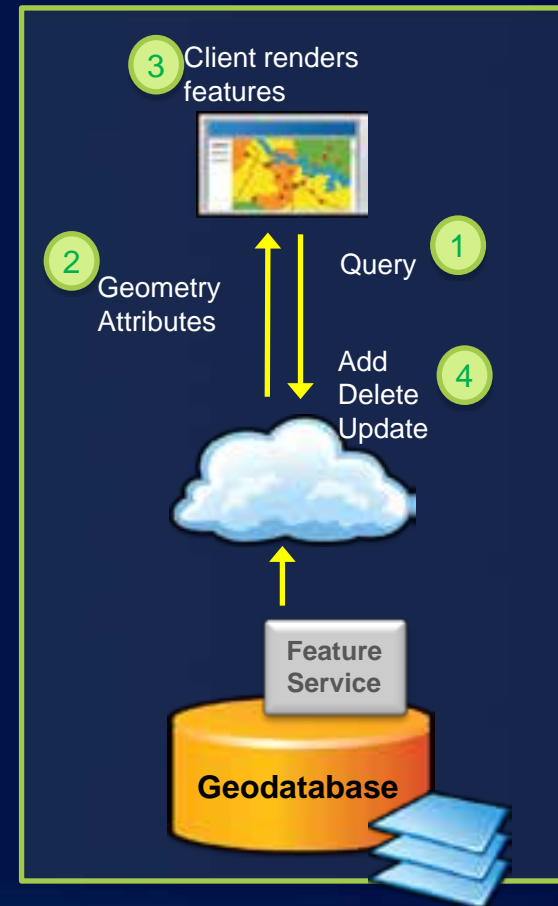
# Feature Services in ArcGIS

*From simple sketching to sophisticated editing through web services*



# What is a Feature Service?

- **Added in ArcGIS 10**
- **Designed for web editing**
  - Push changes into the geodatabase over the internet
  - Feature template-based editing
- **Also excellent for querying**
  - Fetch geometries and attributes
  - Client renders features (Thematic mapping, maptips...)
- **Stateless, quick, RESTful service**



How Feature Services work?  
(Simplified)




# The Services Directory view

The screenshot shows a web browser window displaying the ArcGIS Services Directory for a folder named 'SaveTheBay'. The browser address bar shows the URL: <http://184.72.245.153/ArcGIS/rest/services/SaveTheBay>. The page title is 'Folder: SaveTheBay' and the current version is 10.0. There are two services listed: 'SaveTheBay/SaveTheBay (MapServer)' and 'SaveTheBay/SaveTheBay (FeatureServer)'. A callout box labeled 'Your Map Service' points to the MapServer service. Another callout box labeled 'It's Feature Service' points to the FeatureServer service. A third callout box labeled 'Templates' points to the 'Drawing Info' section of the detailed view on the right. The detailed view shows the following information:

Spatial Reference: 102113

**Drawing Info:**

Renderer:  
Unique Value Renderer:  
Default Label:  
Unique Value Infos:

- Value: Chicks Sighted  
Label: Chicks Sighted  
Description:  
Symbol:  
*Picture Marker Symbol:*  

- Value: Confirmed Sighting  
Label: Confirmed Sighting  
Description:  
Symbol:  
*Picture Marker Symbol:*  

- Value: Nesting Pair  
Label: Nesting Pair  
Description:  
Symbol:  
*Picture Marker Symbol:*  


# How to create a web editing application?

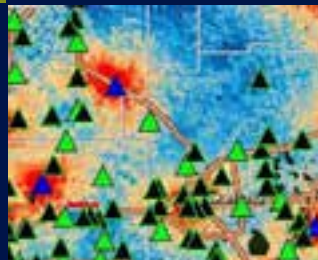


# Feature Service Basics Demo

Fulton County Dept. of Health and Wellness/District 3, Unit 3



- Authoring the Map and Editing Templates
  - Publishing the service
    - Using it





# Web Editing Scenarios



- Attachments
- **Editor tracking**
- Ownership Based Access Control
- Handling complex symbology/geometries
  - Disable geometry edits
  - Editing versions
- Sophisticated Web Editing

MICKLEFIRTH  
CHANNEL

# Web Editing Scenarios



- Attachments
- Editor tracking
- **Ownership Based Access Control**
- Handling complex symbology/geometries
  - Disable geometry edits
  - Editing versions
- Sophisticated Web Editing

MICKLEFIRTH  
CHANNEL

# Web Editing Scenarios



- Attachments
- Editor tracking
- Ownership Based Access Control
- **Handling complex symbology/geometries**
  - Disable geometry edits
  - Editing versions
- Sophisticated Web Editing

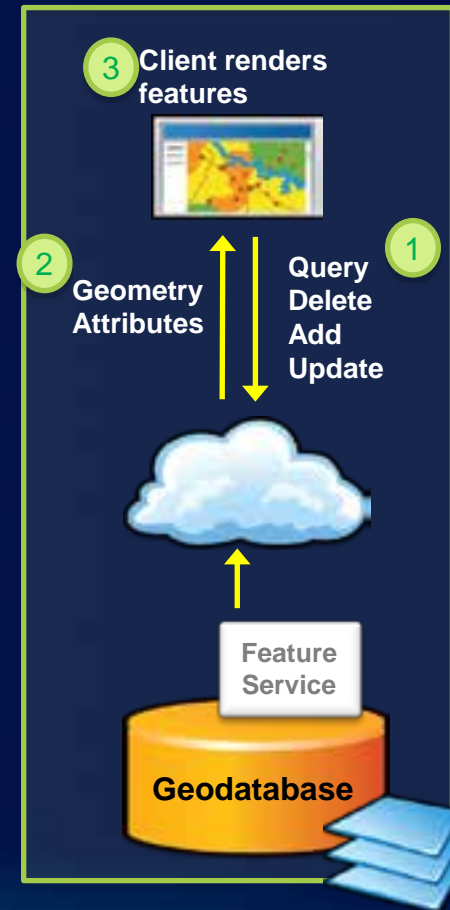
MICKLEFIRTH  
CHANNEL

# Feature Service usage web editing patterns I

Feature Service to edit and graphics render

- Render features in the client (Feature Layer)
  - Fetch all at once (Snapshot mode)
  - Or as needed (On demand mode)
- 
- Subset of ArcGIS symbology
  - No more than a few hundred features in display\*
  - Careful with large polys/polylines
  - Take advantage of maptips (popups)

\* Be smart about using scale dependencies, generalize, filter...

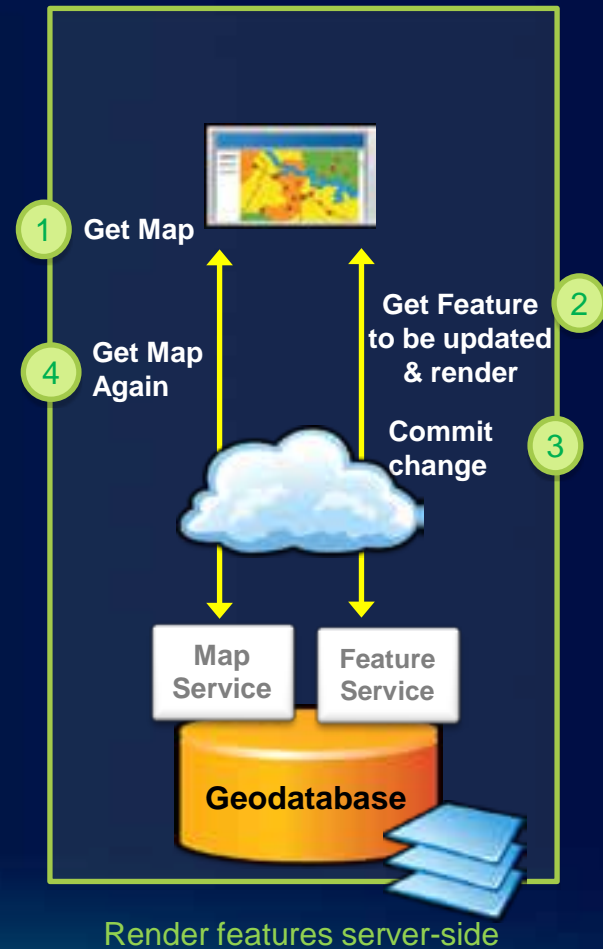


Render features client-side

# Feature Service usage web editing patterns II

## Feature Service to edit and map service to render

- Render features in the server (Map service)
- Refresh map after every update
- Features being edited displayed in client
  - Through FeatureLayer Selection mode
- Full symbology
- Many and complex features



# Web Editing Scenarios



- Attachments
- Editor tracking
- Ownership Based Access Control
- Handling complex symbology/geometries
  - **Disable geometry edits**
- Editing versions
- Sophisticated Web Editing

MICKLEFIRTH  
CHANNEL

# Web Editing Scenarios



- Attachments
- Editor tracking
- Ownership Based Access Control
- Handling complex symbology/geometries
  - Disable geometry edits
  - **Editing versions**
- Sophisticated Web Editing

MICKLEFIRTH  
CHANNEL

# Web Editing Scenarios



- Attachments
- Editor tracking
- Ownership Based Access Control
- Handling complex symbology/geometries
  - Disable geometry edits
  - Editing versions
- **Sophisticated Web Editing**

MICKLEFIRTH  
CHANNEL

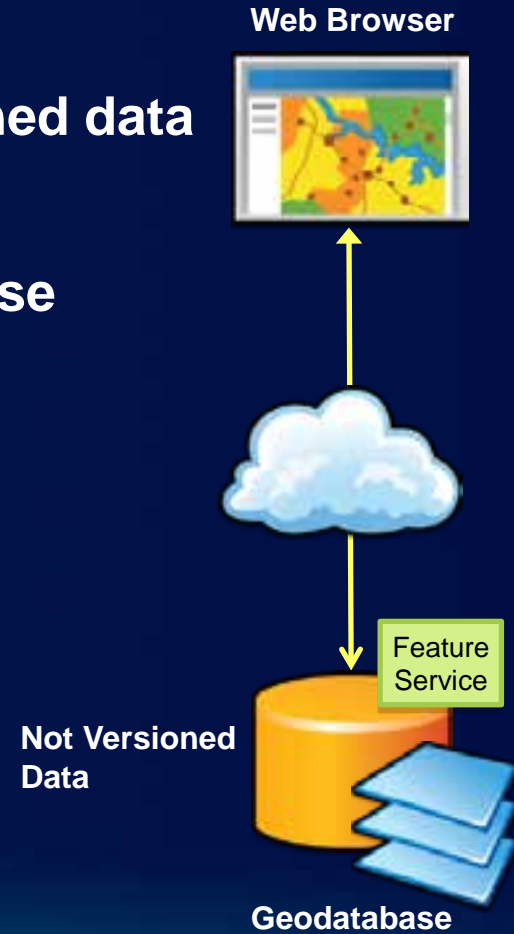


# Sophisticated Web Editing

- **Edit toolbar available with the web APIs**
- **Geometry Service**
  - **ArcGIS Server service for geometry manipulation**
- **Server Object Extensions**
  - **how you extend ArcGIS Server**
- **ArcGIS desktop**
  - **Tools to edit and sync a local copy**

# Editing native spatial types in databases

- Feature service against non versioned data
- Last-in wins
- Changes directly applied on database
- Use Spatial Data Server



# The challenge is the user experience

- **Adding many tools is an easy and often unnecessary part**
- **Build focused editing apps**
  - The necessary tools, no more
  - The workflow drives the design, not the opposite!
- **Start by selecting the right client**
  - Do not reinvent the wheel
  - Do not push the technology beyond its comfort level
  - Or your users!
- **Develop if needed**

# Summary

- **ArcGIS Server features built-in web editing capabilities**
- **Enabling many exciting applications**
  - **Crowd sourcing, geocollaboration, web editing...**
- **Feature Services**
- **Out of the box editing clients**
  - **ArcGIS.com Viewer**
  - **ArcGIS Viewer for Flex**
- **ArcGIS Web Mapping APIs**
  - **FSVersionedEditing app on ArcGIS.com**
- **Geometry Service and Server Object Extensions**

# Presentation Outline

- Introduction
- Databases and Geodatabases
- ArcObjects
- SQL
- Python
- File Geodatabase API
- Runtime
- Web Editing
- **Questions and other information**

# Additional Resources

- ***Meet the Teams***
  - *Tuesday at 6:00 – 7:30 pm*
  
- **ESRI Showcase**
  - **Monday, March 26, 11:00 a.m.–7:30 p.m.**
  - **Tuesday, March 27, 12:30 p.m.–6:00 p.m.**
  - **Wednesday, March 28, 10:00 a.m.–6:00 p.m.**
  
- ***ESRI Resource Centers***
  - **PPTs, code and video**

