# Effective Geodatabase Programming

Craig Gillgrass

# Assumptions

- Basic knowledge of SQL, Python and relational databases
- Basic knowledge of the Geodatabase
- We'll hold all questions till end

Please turn off cell phones

# ArcGIS Is a Platform

## Enabling Web GIS Everywhere

- Simple
- Integrated
- Open


*Available in the Cloud . . .*
*. . . and On-Premises*

# 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

# 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

Parcels

| OBJECTID * | SHAPE * | PROPERTY_I * | Res | Zoning_simple | SHAPE_Length | SHAPE_Area |
|---|---|---|---|---|---|---|
| 1 | Polygon | 5001 | Non-Residential | <Null> | 3597.760813 | 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 | 521.761248 | 14036.135346 |
| 5 | Polygon | 1005 | Residential | Residential | 453.479649 | 9816.352665 |

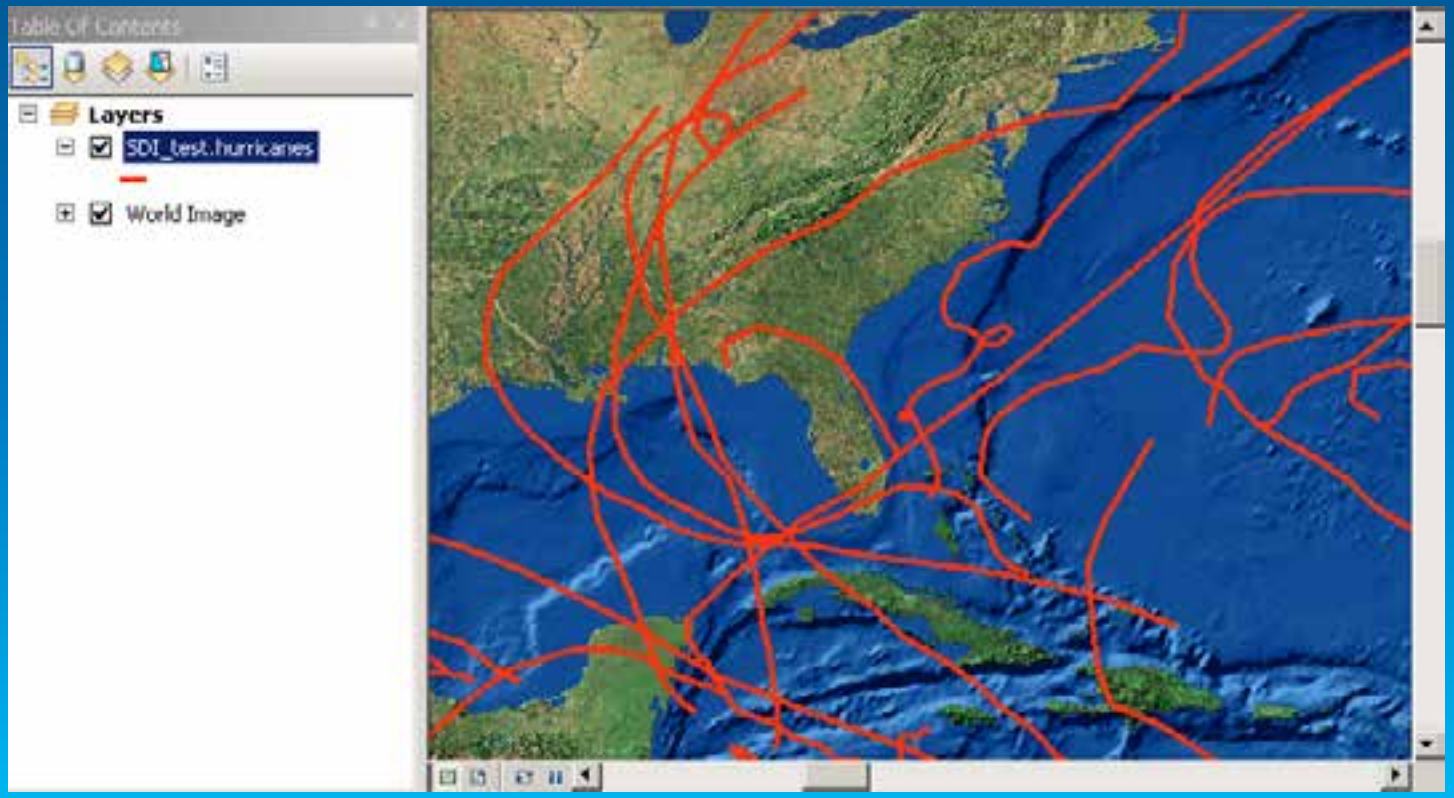
# Viewing database data in ArcGIS

- 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

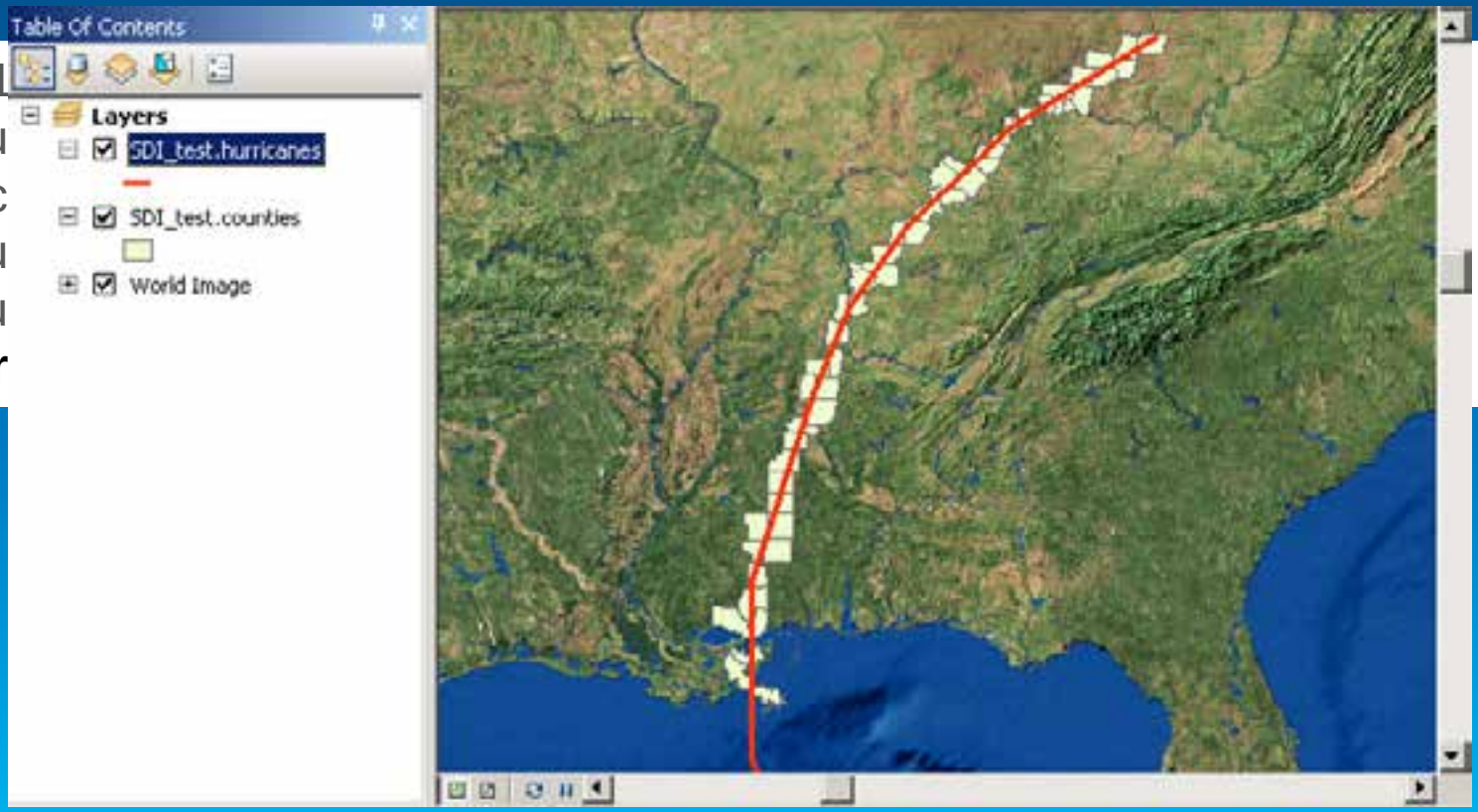
- **Most complex SQL query that uses casting, derived columns and spatial operators**

```
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

## Cases where you want to do more with your data

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

# The Geodatabase

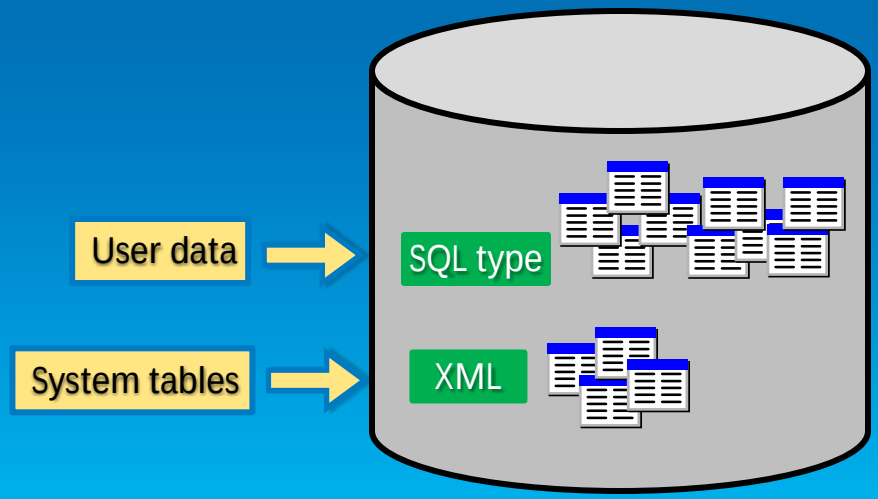
- A physical store of geographic data
  - Scalable storage model supported on different platforms
- Core ArcGIS information model
  - A comprehensive model for representing and managing GIS data
  - Implemented as a series of simple tables
- A transactional model for managing GIS workflows
- APIs for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

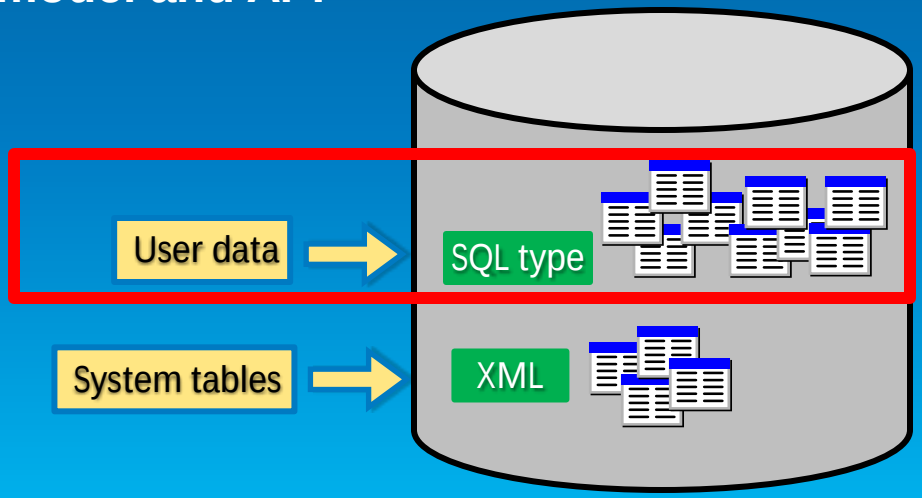
# Geodatabase Schema

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



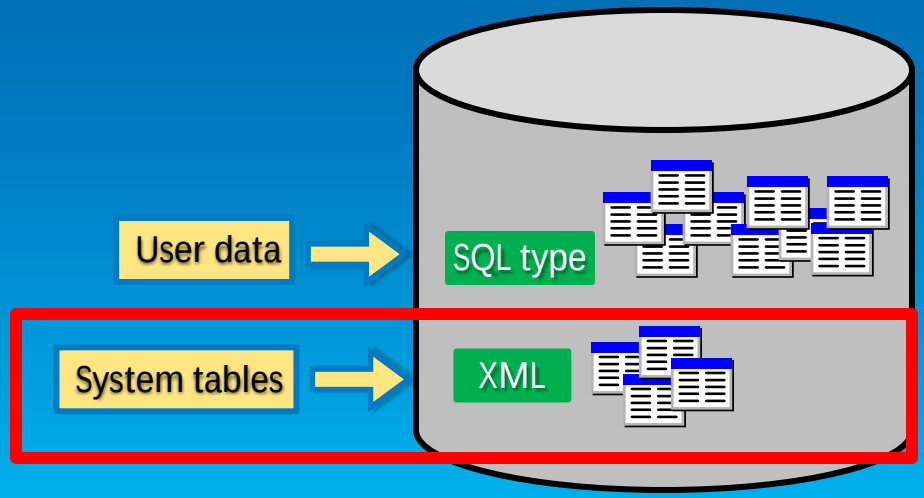
# User-defined tables

- 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

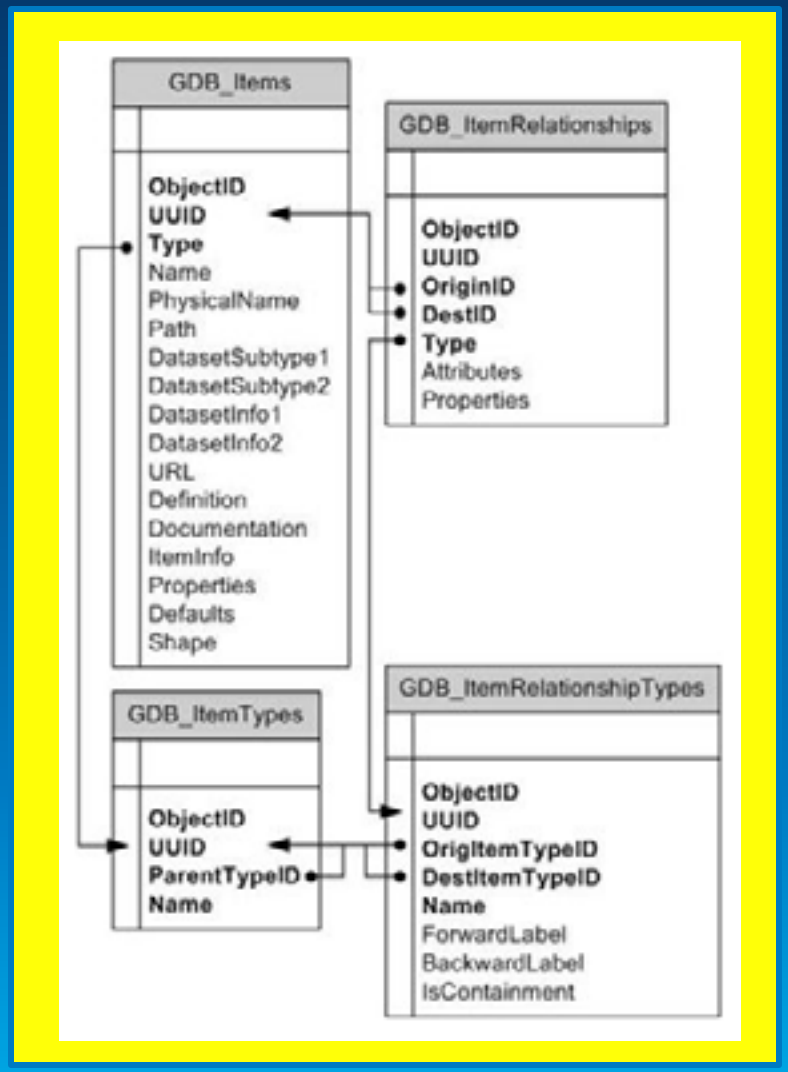


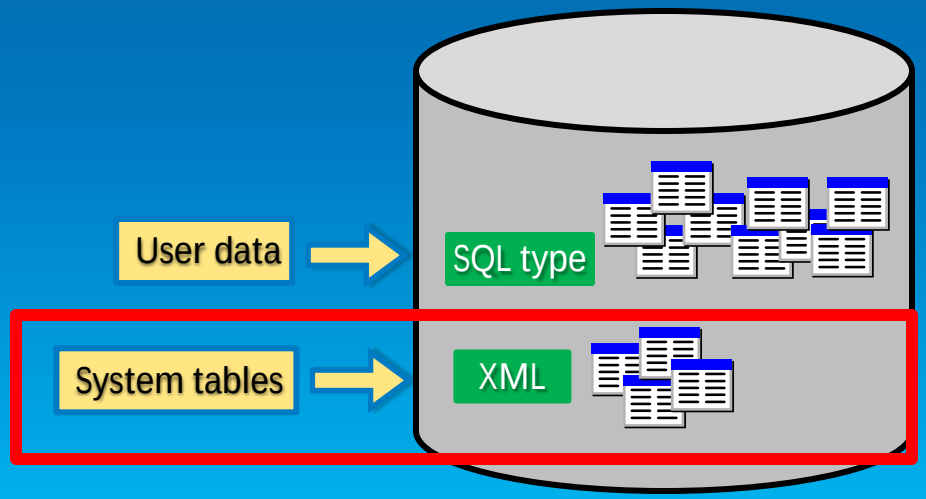
# Geodatabase system tables

- 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

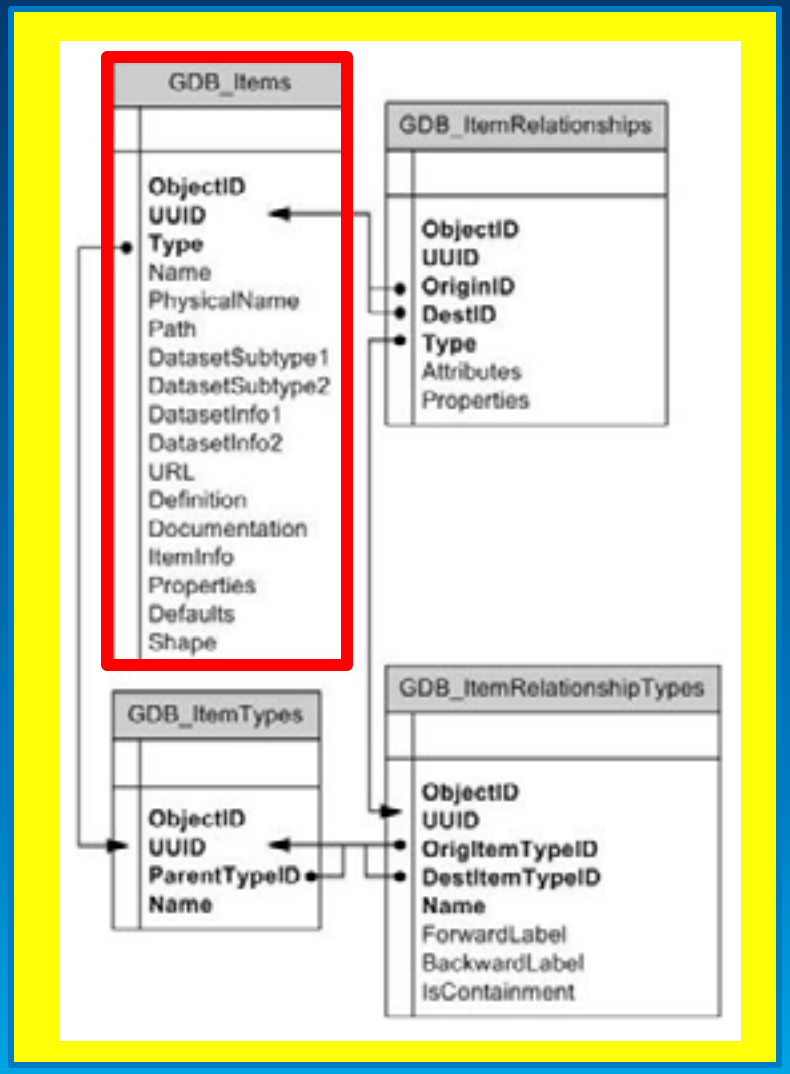


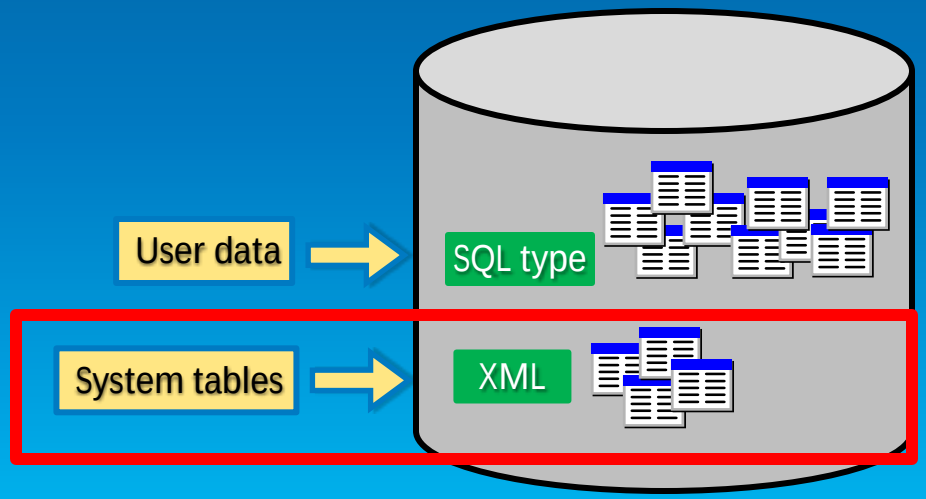
# Geodatabase Schema…



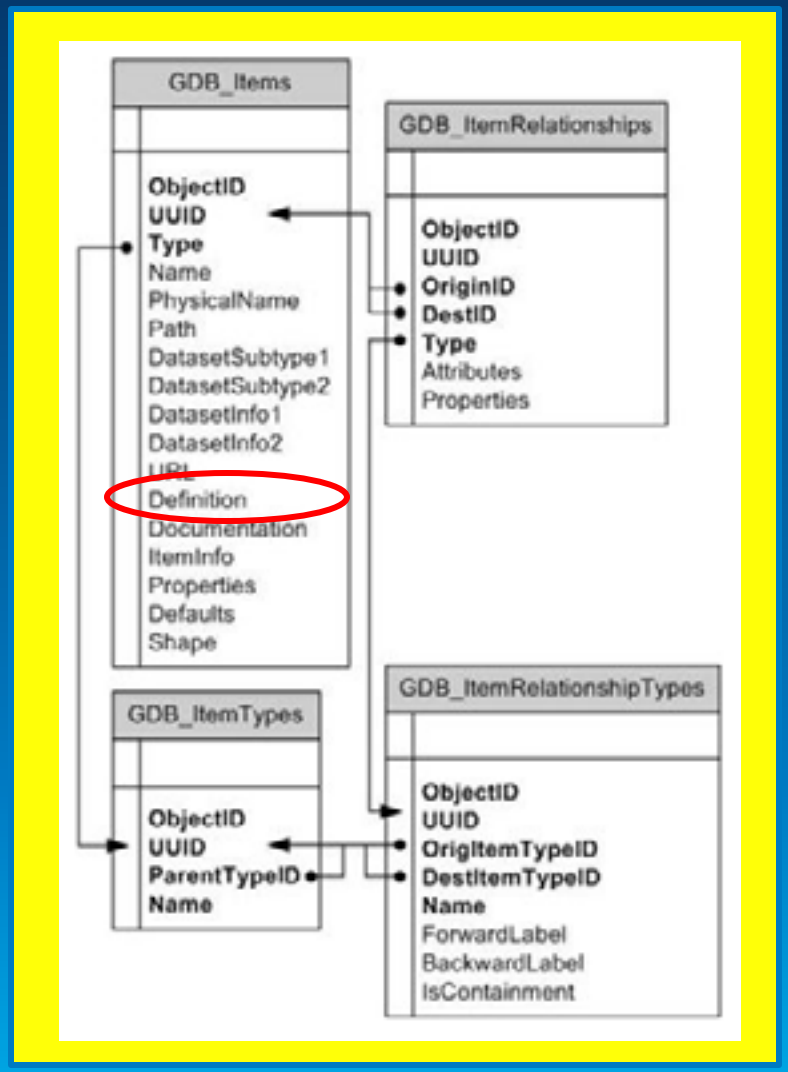



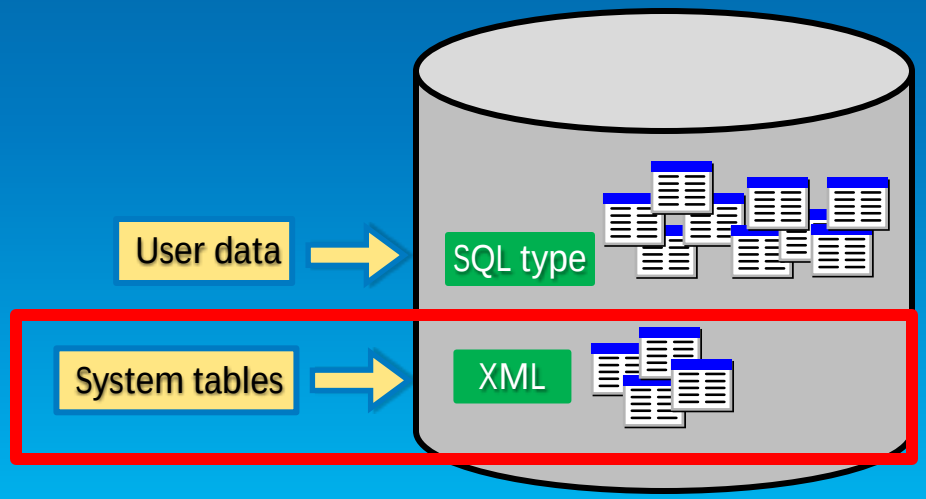
# Geodatabase Schema…



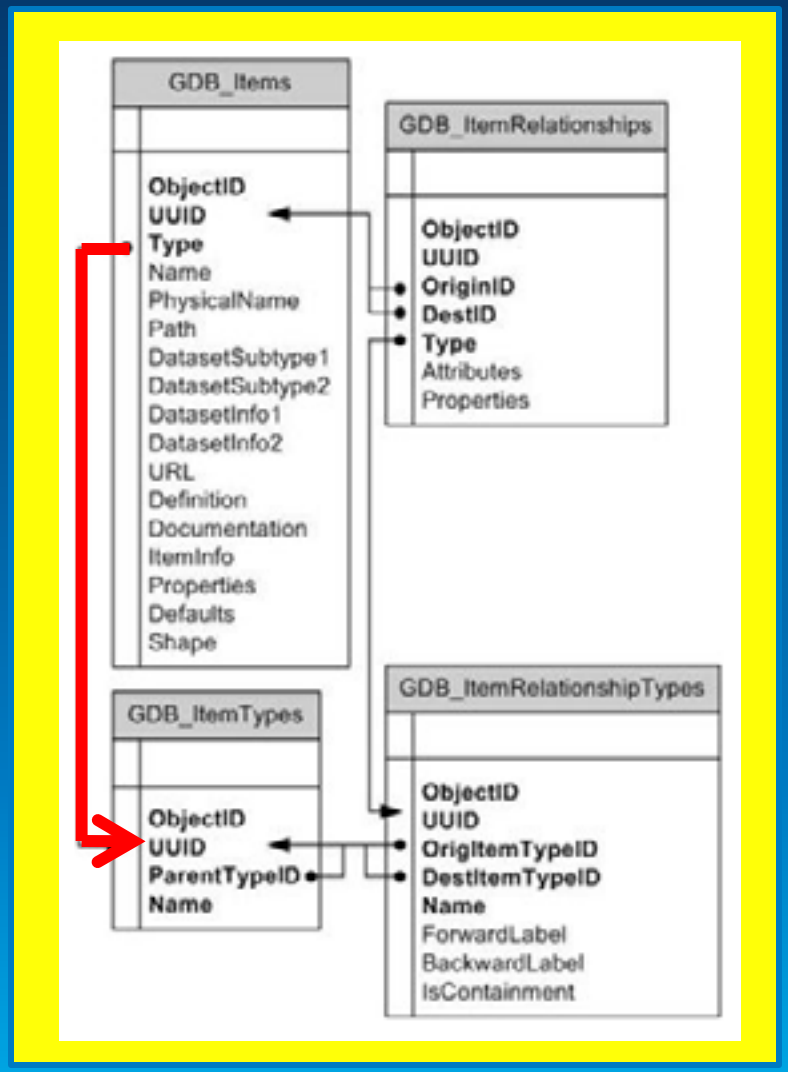



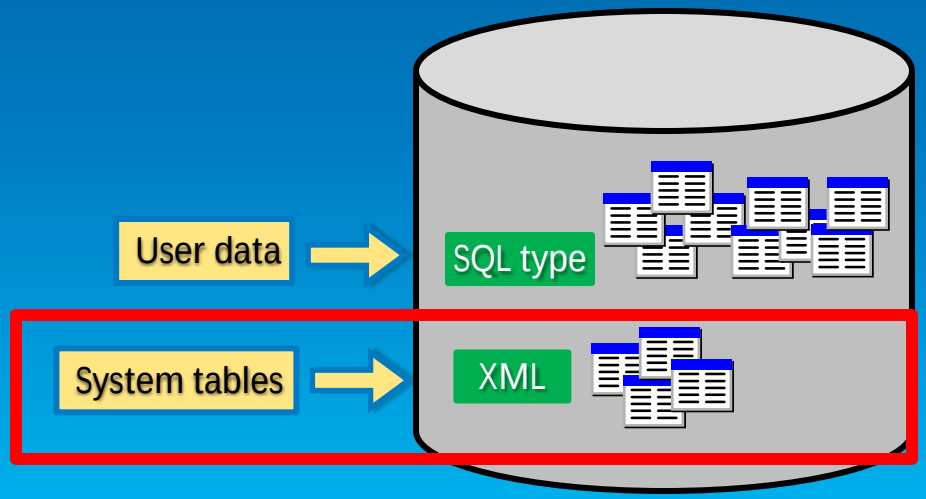
# Geodatabase Schema…



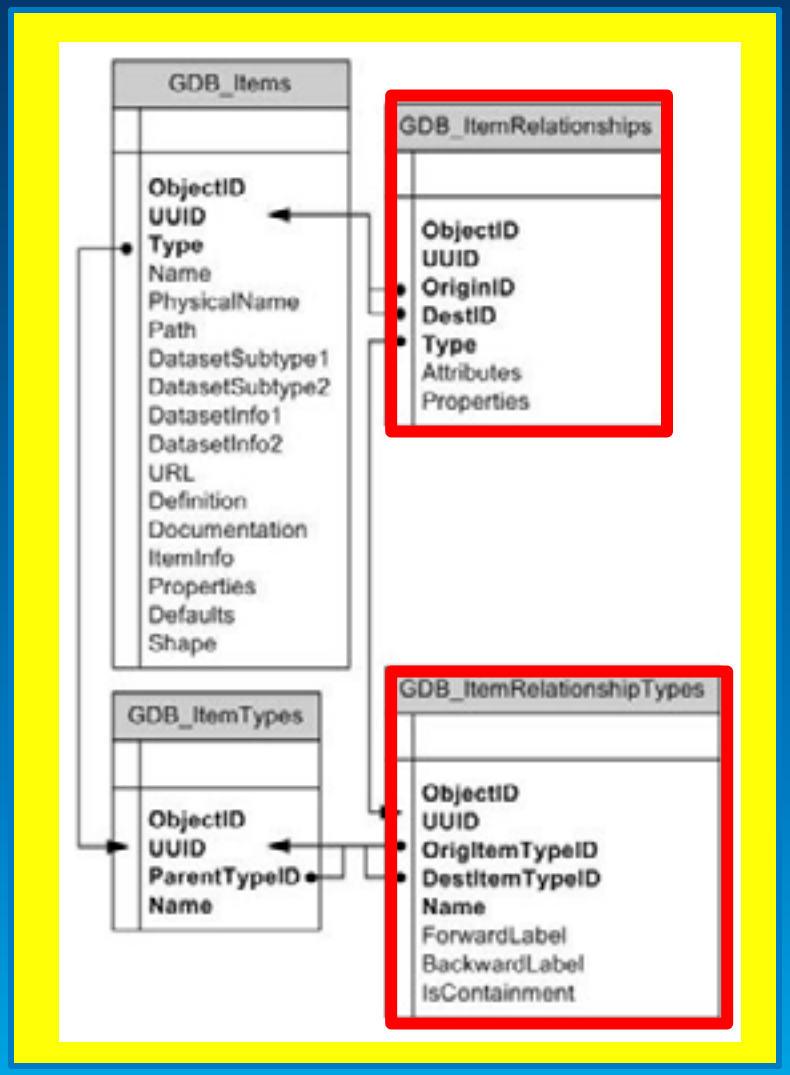



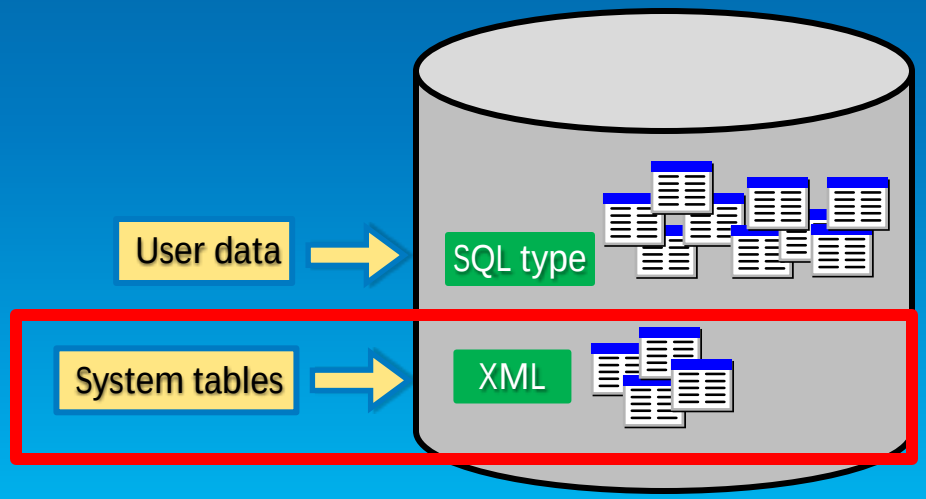
# Geodatabase Schema…



# Geodatabase Schema…



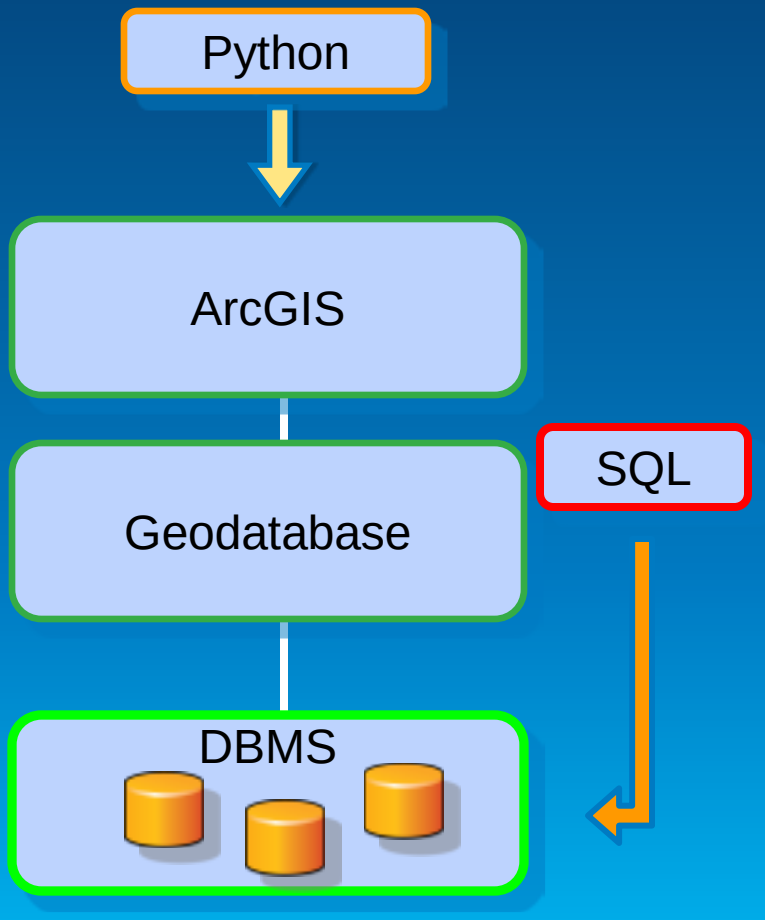



# Accessing Geodatabase through SQL

- Access schema and properties of existing datasets
  - Use SQL statements to query the *definition* attribute on the *gdb_items* table
- Editing tables/feature classes, whether versioned or not
  - Via versioned views with versioned classes
- Create tables with SQL containing spatial or raster types
- Leverage SQL functions to evaluate attributes and spatial relationships, perform spatial operations, and return and set spatial properties

# Accessing Geodatabase through SQL

- With SQL, you access 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…



Demo

# Accessing a geodatabase through SQL

- Resolving
  - Coded Value Domains
  - Feature Dataset Relationships
  - Domain References

# 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**

# What are the benefits of a spatial type?

- Using SQL with a spatial type you can
  - Create tables with a spatial attribute
  - Read and analyze the spatial data
  - Insert, update, and delete simple geometry data

Spatial Type

SQL

Parcels

| OBJECTID * | SHAPE * | PROPERTY_I * | Res | Zoning_simple | SHAPE_Length | SHAPE_Area |
|---|---|---|---|---|---|---|
| 1 | Polygon | 5001 | Non-Residential | <Null> | 3597.760813 | 112552.418591 |
| 2 | Polygon | 5002 | Non-Residential | <Null> | 814.855837 | 18488.417709 |
| 3 | Polygon | 1003 | Residential | Residential | 489.655523 | 12815.591379 |
| 4 | Polygon | 1004 | Residential | Residential | 521.761248 | 14036.135346 |
| 5 | Polygon | 1005 | Residential | Residential | 453.479649 | 9816.352665 |

# Accessing Geodatabase through SQL

- Can use SQL to create, insert and update tables
  - Need to register the table with the geodatabase to participate in geodatabase functionality

```
CREATE TABLE hazardous_sites
  (oid INTEGER NOT NULL, site_id INTEGER,
   name VARCHAR(40), location sde.st_geometry)
```

- Cannot modify schema of registered tables (i.e add a field) or create geodatabase items (i.e domains) through SQL

# Accessing Geodatabase through SQL

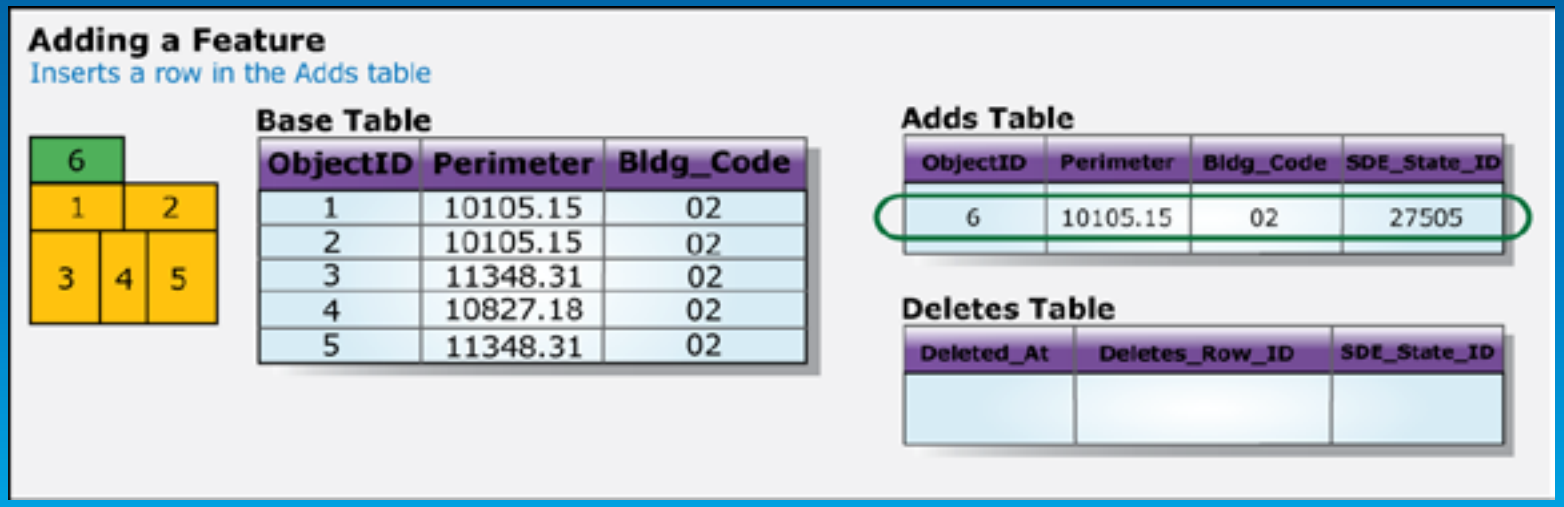
- Editing feature classes with SQL and spatial type
  - Simple features (Points, lines, polygons)
  - Without geodatabase behavior
  - Use the **Is_Simple** function to determine whether your data can be updated
- Editing tables/feature classes
  - Use SQL SELECT statements
  - Directly editing the database tables (no delta tables)
  - Non-versioned editing in ArcGIS terminology
- Editing versioned tables/feature classes
  - Requires versioned views

# Editing tables/feature classes

- 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
- 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

- Changes tracked on delta tables (Adds and Deletes tables)
- Support concurrent editing with long transactions (hours/days)
- Undo/redo editing experience
- No locking or data extraction required



# Editing versioned tables and feature classes

- Use versioned views
- Must use several stored procedures/commands installed with the geodatabase
  - Create versioned 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 non-versioned editing, ObjectID values for new records are automatically generated
  - Changes are made to the delta tables
  - Versions must be reconciled through ArcGIS

Demo

# Accessing a geodatabase through SQL

- **Editing**
  - **Versioned and Non Versioned Classes**
  - **Working with Views**

# Second Half Agenda

- Why use Python?
- Tips for using python with geodatabases
- Demo: Creating geodatabases and schema
- Demo: Performing geodatabase maintenance

# Why use Python for Administration?

- Numerous tools available
  - Schema creation and administration
  - Maintenance
- Cross Platform
- Easy to schedule tasks

# Using Python to access your geodatabase

- Connection files
  - Create Database Connection tool
- Version access is defined in the connection file.
- Connected user is defined in the connection file.
- Multiple connections = multiple connection files.

Demo

# Creating a Geodatabase

# Demo 1: Creating a geodatabase

- Create an enterprise geodatabse
- Create database roles
- Create users
- Create schema
- Apply privileges
- Register data as versioned
- Create edit versions

Demo

# Performing Maintenance

# Demo 2: Geodatabase maintenance

- Blocking and accepting connections
- Disconnecting users
- Reconcile/post versions
- Compress geodatabase
- Updating statistics and indexes
- Email notifications
- Scheduling

# Summary

- GDB is open to SQL/Python Devs
- Through SQL use XML field in the GDB_Items table
- Can also edit data through SQL
- Schema creation/admin with Python
- GDB administration with Python
- Automate most GIS processes