Esri Developer Summit

March 26–29, 2012 | Palm Springs, California

esri.com/events/devsummit

Accessing and Administering your Enterprise Geodatabase through SQL and Python

Brent Pierce @brent_pierce

Russell Brennan @russellbrennan

hashtag: #sqlpy

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

What is 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
  - Relational integrity
  - Reliability, Flexibility, Scalability
  - Supports continuous, large datasets
  - Standard relational database schema
  - Base short transaction model
  - Supports structured query language (SQL)

Geodatabase is based on relational principles …

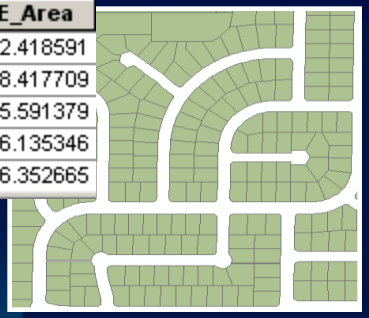
- Leverages key DBMS principles and concepts to store geographic data as tables in a DBMS
  - Data is organized into tables
  - Tables contain rows
  - All rows in a table have the same attributes
  - Each attribute has a type
  - Relational integrity rules exist for tables

Geodatabase is based on relational principles …

- A feature class is stored as a simple DBMS table
- 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>	3597.780813	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

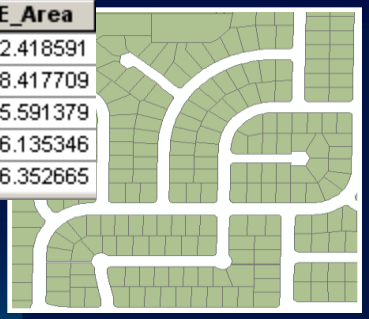


Geodatabase is based on relational principles …

- A feature class is stored as a simple DBMS table
- 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

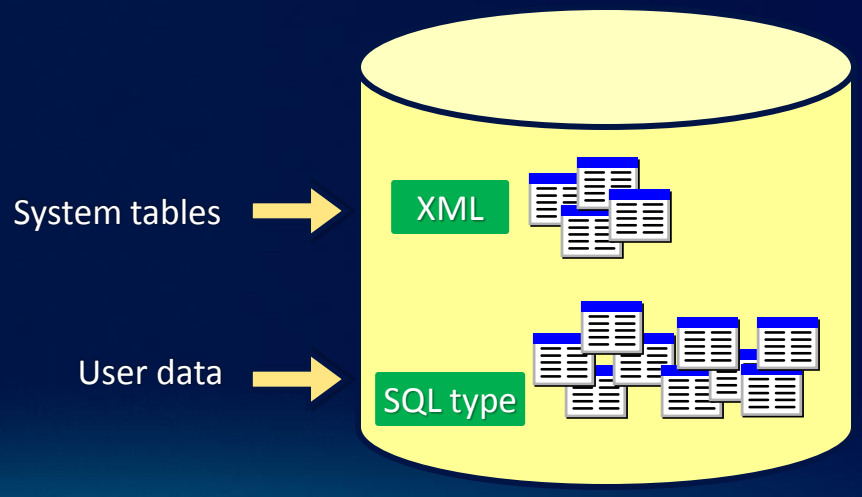
Parcels

OBJECTID *	SHAPE *	PROPERTY_I *	Res	Zoning_simple	SHAPE_Length	SHAPE_Area
1	Polygon	5001	Non-Residential	<Null>	3597.780813	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



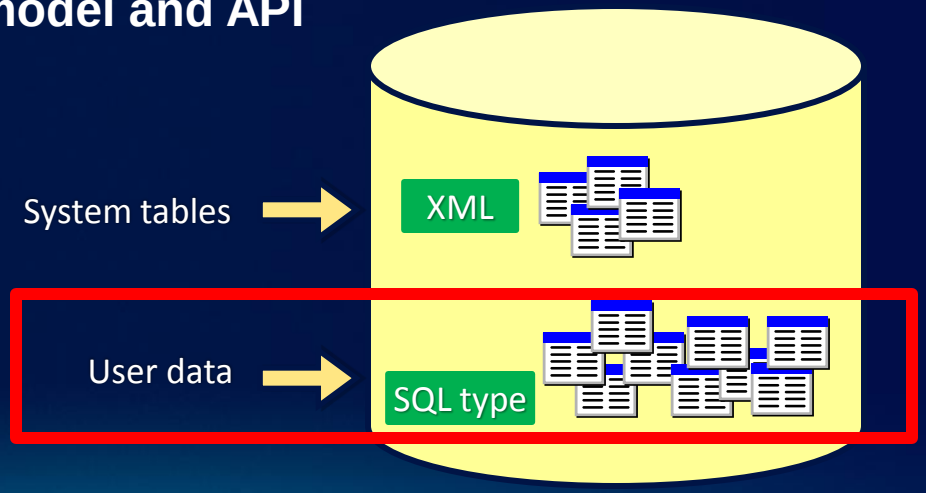
Geodatabase Schema

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



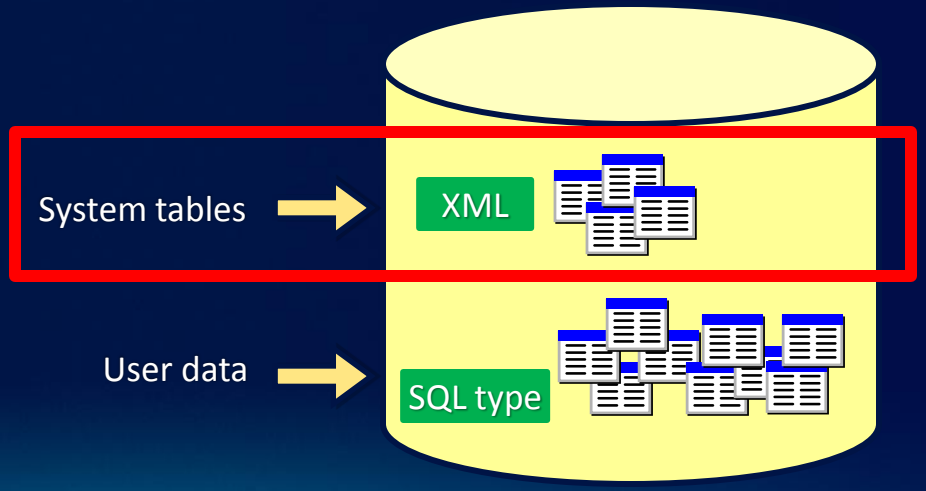
User-defined tables

- 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

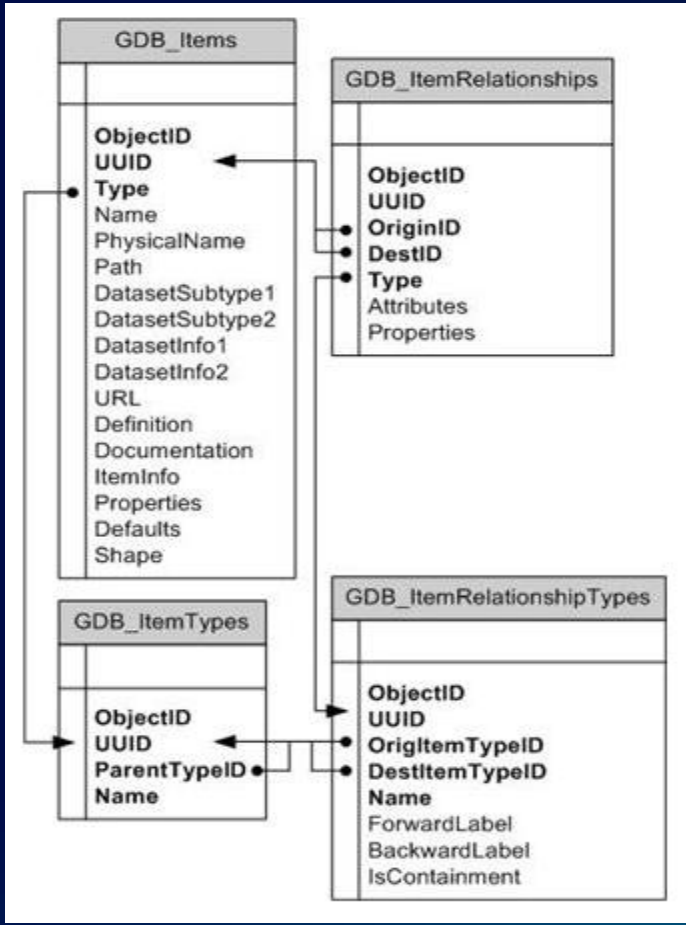


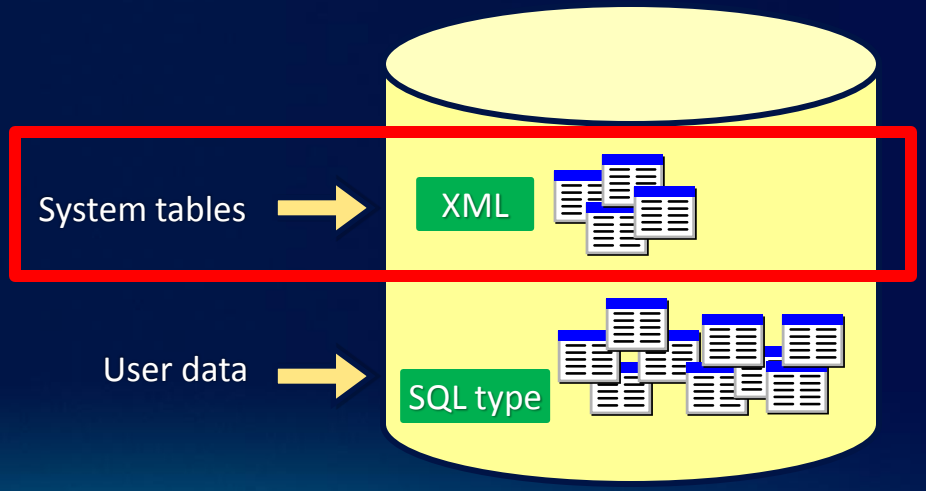
Geodatabase system tables

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

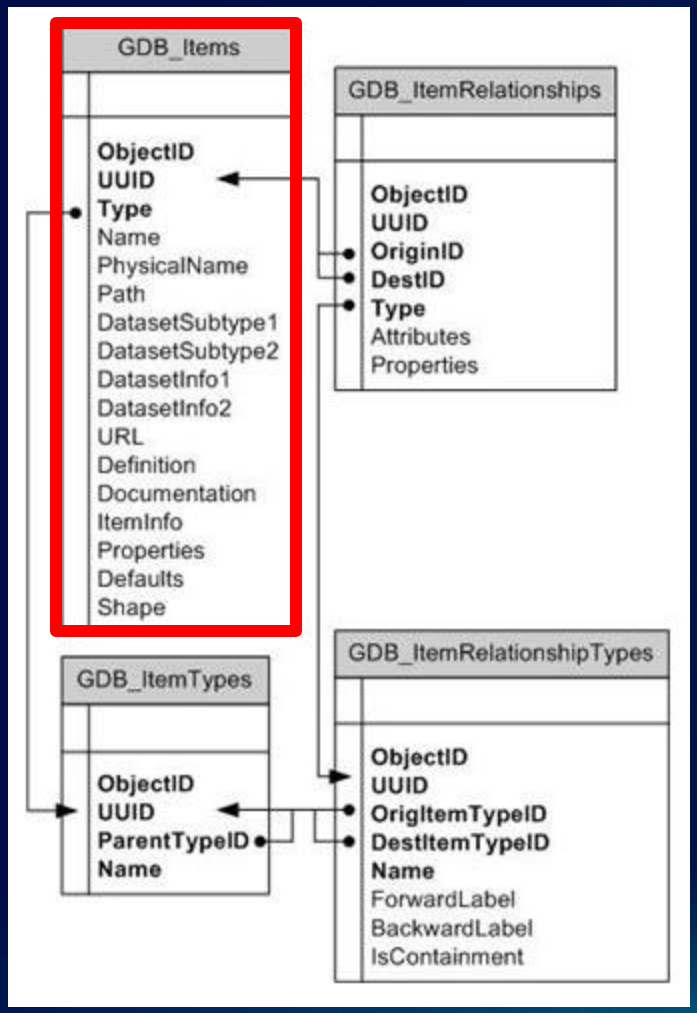


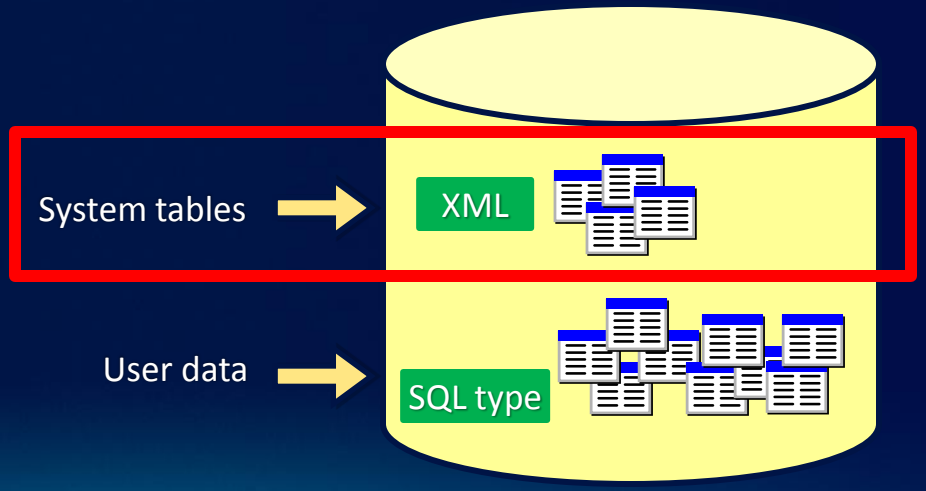
Geodatabase Schema…



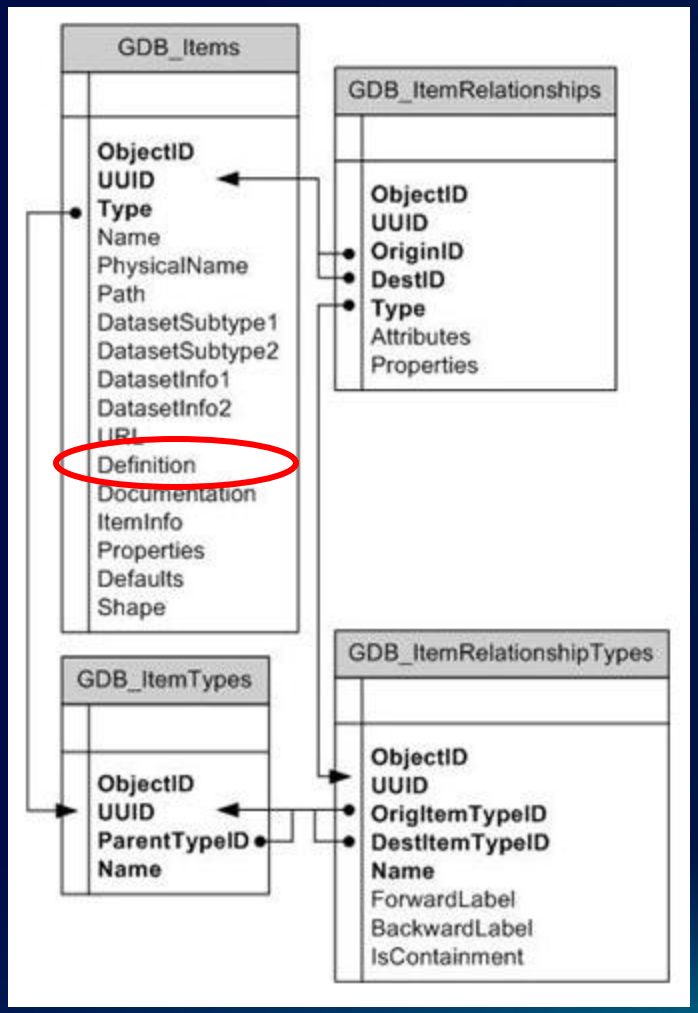


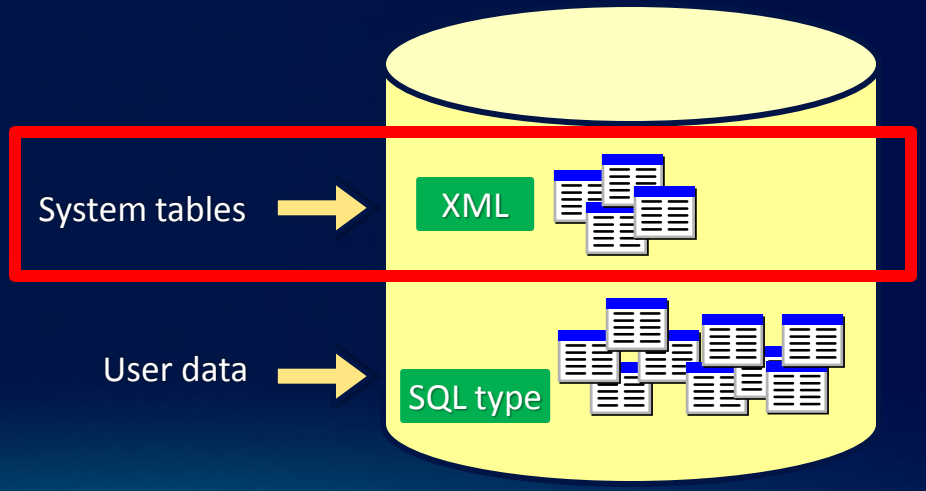
Geodatabase Schema…



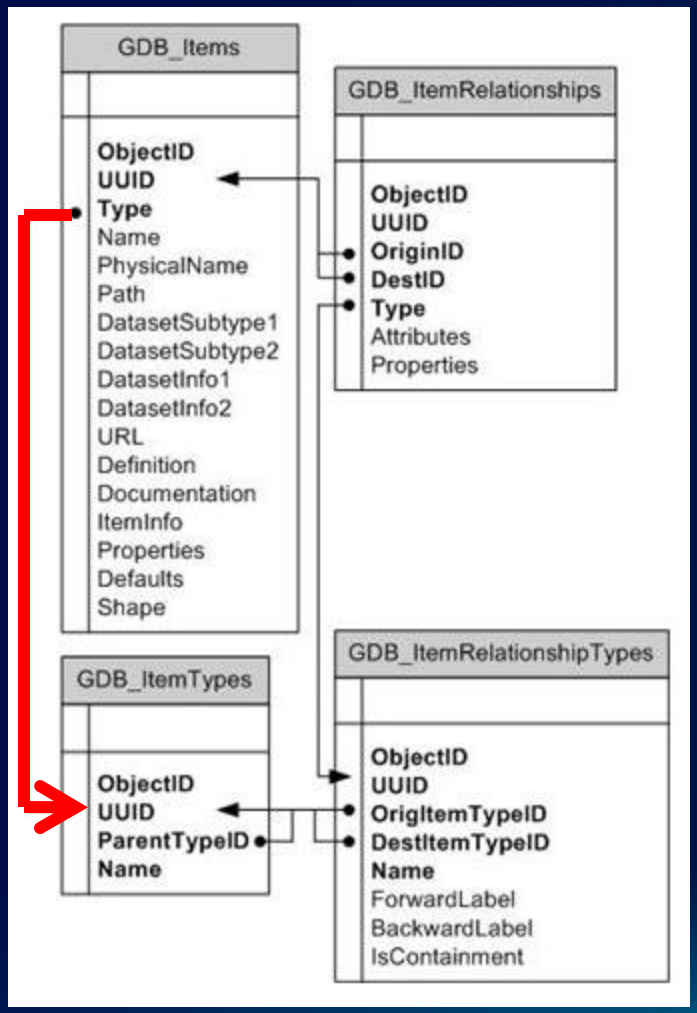


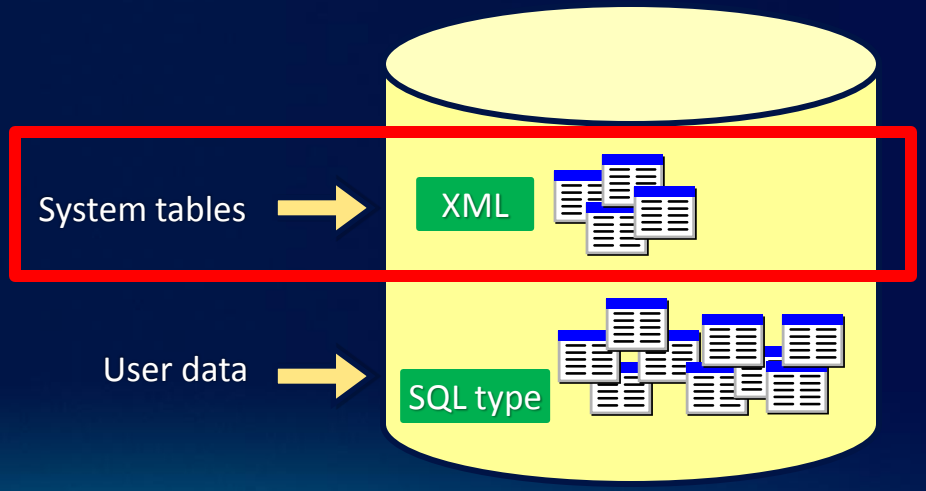
Geodatabase Schema…



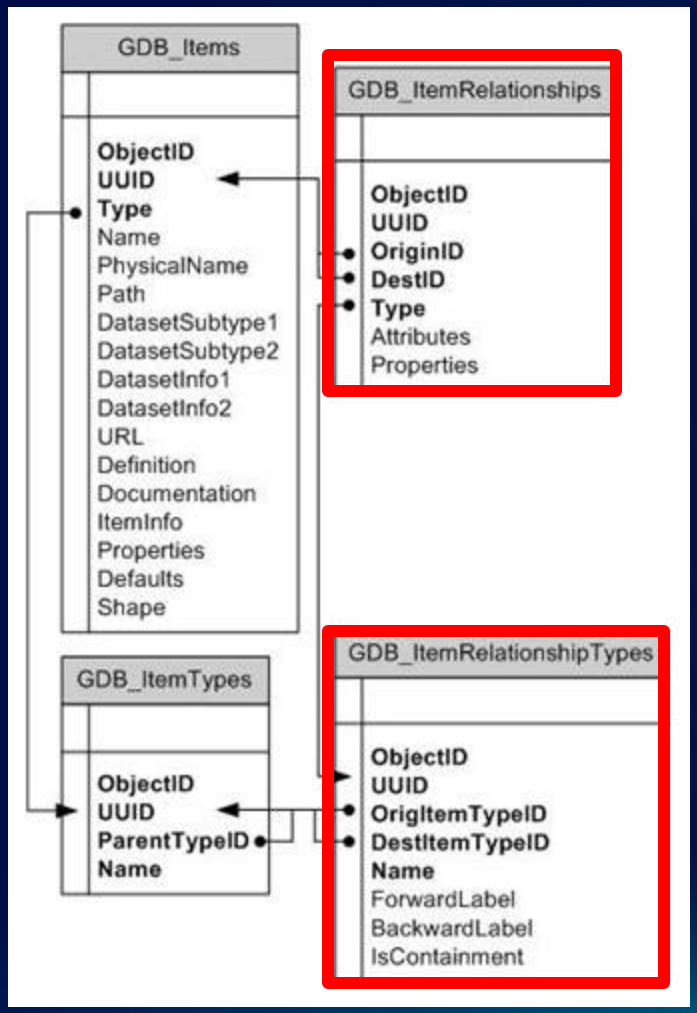


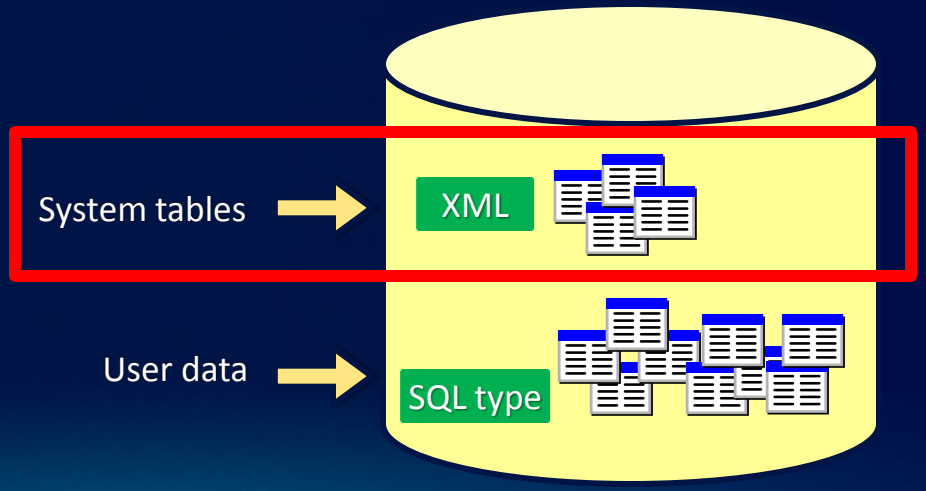
Geodatabase Schema…





Geodatabase Schema…





What is a spatial type?

- A spatial type (ST_Geometry) 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
  - Adheres to standards for SQL access

What are the benefits of a spatial type?

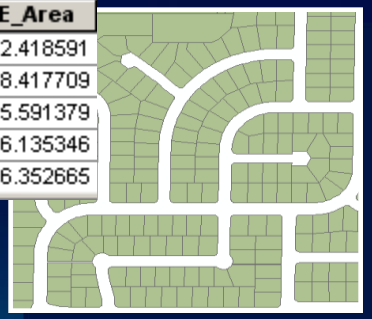
- Using SQL with a spatial type you can
  - Create tables with a spatial attributes
  - Read and analyze the spatial data
  - Insert, update, and delete simple features

Spatial Type

SQL

Parcels

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	489.655523	12815.591379
4	Polygon	1004	Residential	Residential	521.761248	14036.135346
5	Polygon	1005	Residential	Residential	453.479649	9816.352665

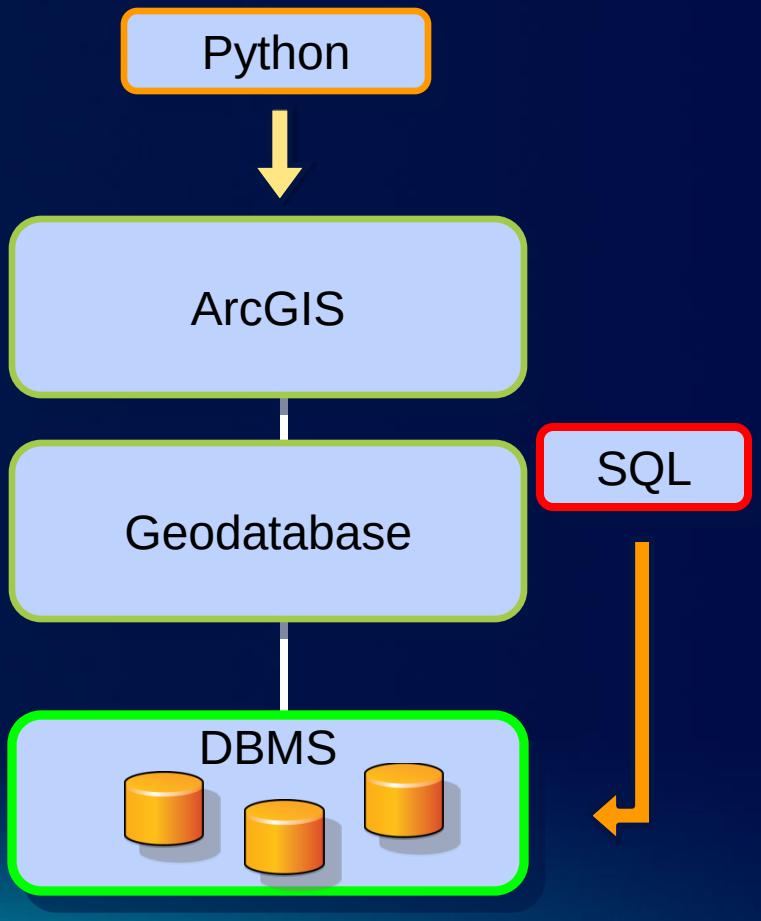


Accessing Geodatabase through SQL

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

- One 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
  - 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 statements
  - Directly editing the database tables (no delta tables)
  - Nonversioned 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

- Use versioned views
  - Versioned Views are automatically created when class is registered as versioned
- Perform edits within the new version

```
SELECT sde.sde_edit_version('WorkOrder1701',1);
```

  - 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

Geodatabase Administration with Python

Russell Brennan

Second Half Agenda

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

Why use Python for Administration?

- Numerous tools available
  - Schema creation and administration
  - Maintenance
- Cross platform.
- Easy access to complex behavior.
- 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 geodatabase.
- 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 database
- Updating statistics and indexes
- Email notifications
- Scheduling

Demo

Publishing

Publishing

- Creating a script tool
- Publishing using the geoprocessing framework
- Consuming the tool

Slide Deck

http://www.slideshare.net/brentpierce/enterprise-geodatabase-sql-access-and-administration

Thanks for attending
Questions?

Please fill out session surveys

www.esri.com/sessionevals