

2011 Esri Developer Summit

Palm Springs, CA

Accessing Your Geodatabase Outside of ArcObjects

Craig Gillgrass & Thomas Brown



Overview

- **What is the Geodatabase?**
 - Geodatabase is based on relational principles
 - Geodatabase Schema
- **Spatial Types and the Geodatabase**
- **Accessing Geodatabase through SQL**

Overview

- **Presumed knowledge**
 - **Relational databases**
 - **Geodatabases**
 - **SQL**

- **Discussing capabilities of ArcGIS 10.0**

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**
 - **Support 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**
- **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 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 ...

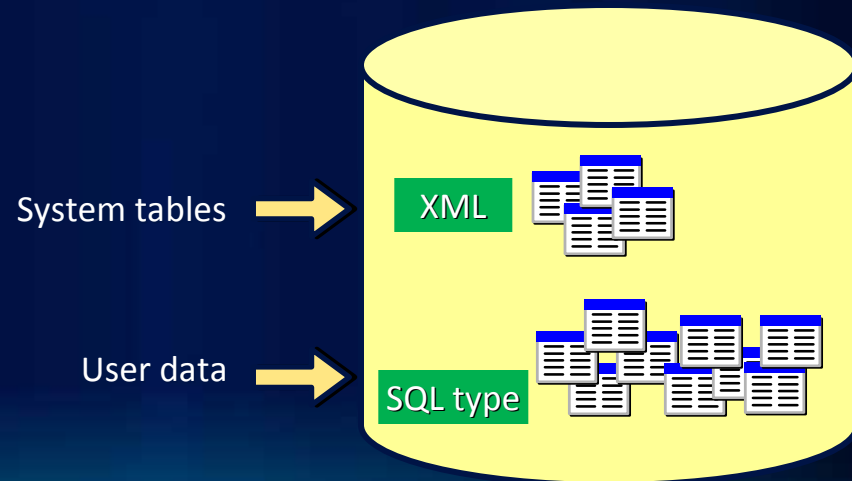
- A feature class is stored as a simple DBMS table
- Each row represents a feature
- The fields in each row represent various characteristics or properties of the feature
- One of the fields holds the feature geometry which is stored as a spatial type

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



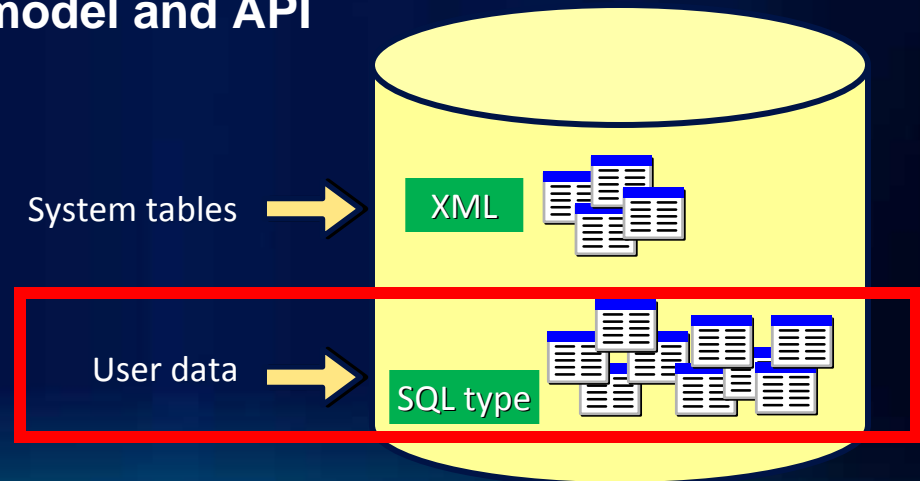
Geodatabase Schema

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



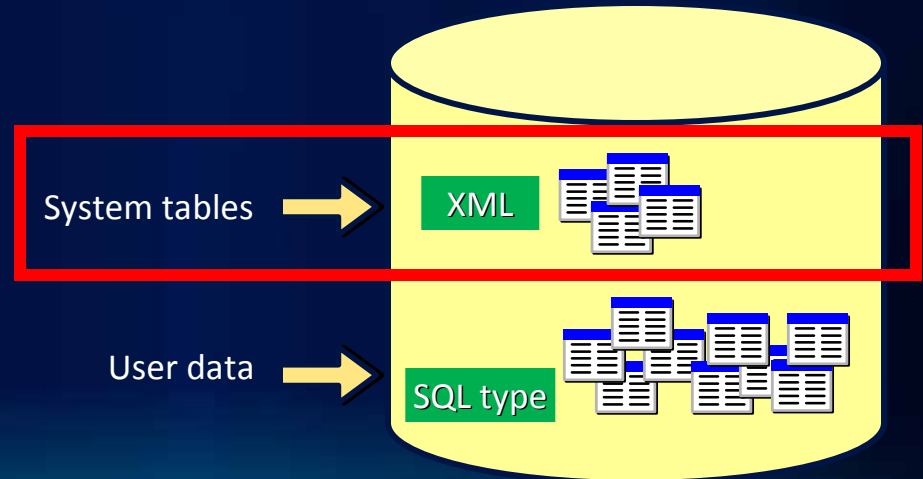
User-defined tables

- 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

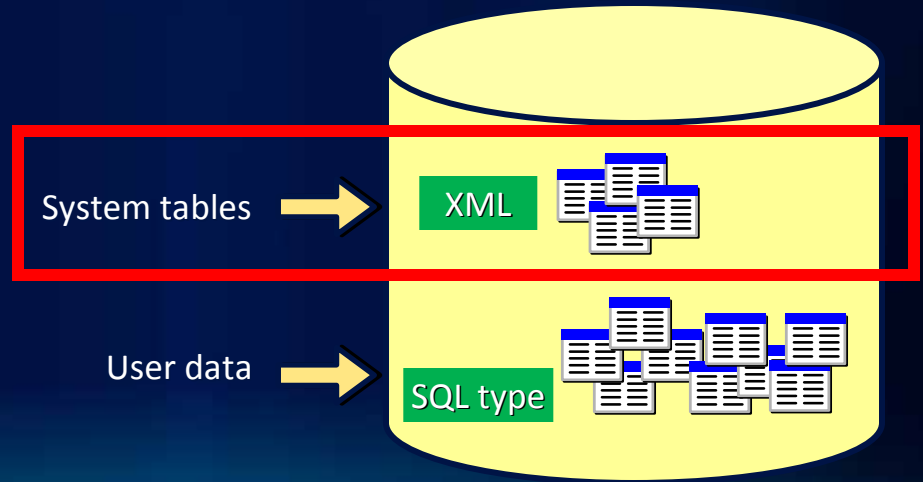
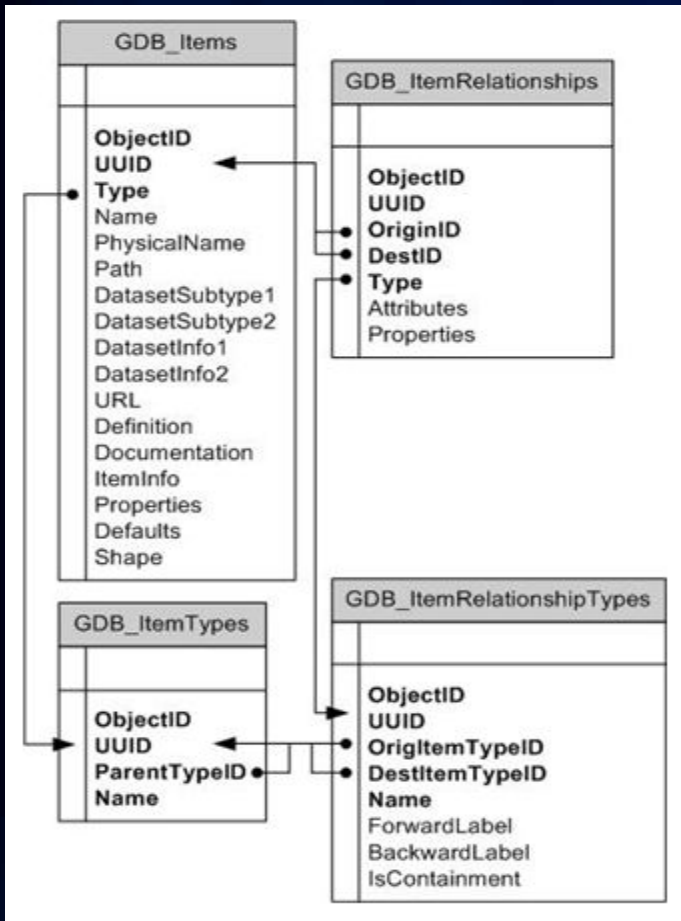


Geodatabase system tables

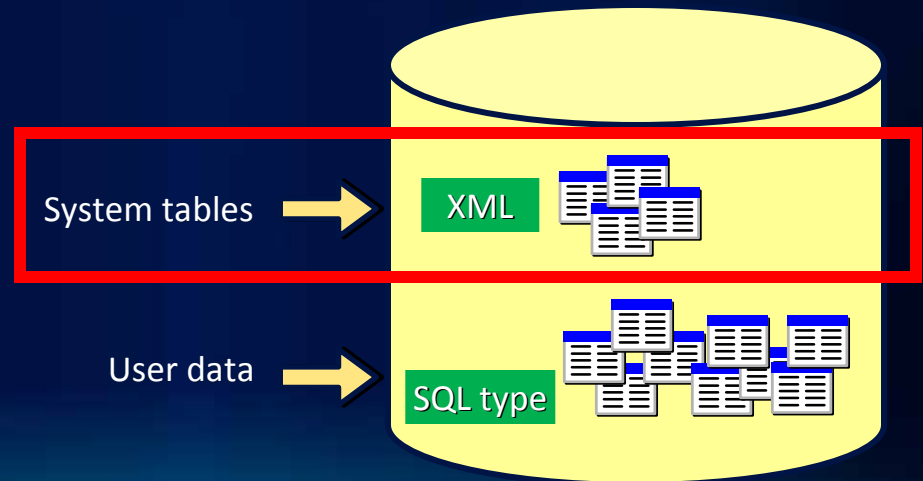
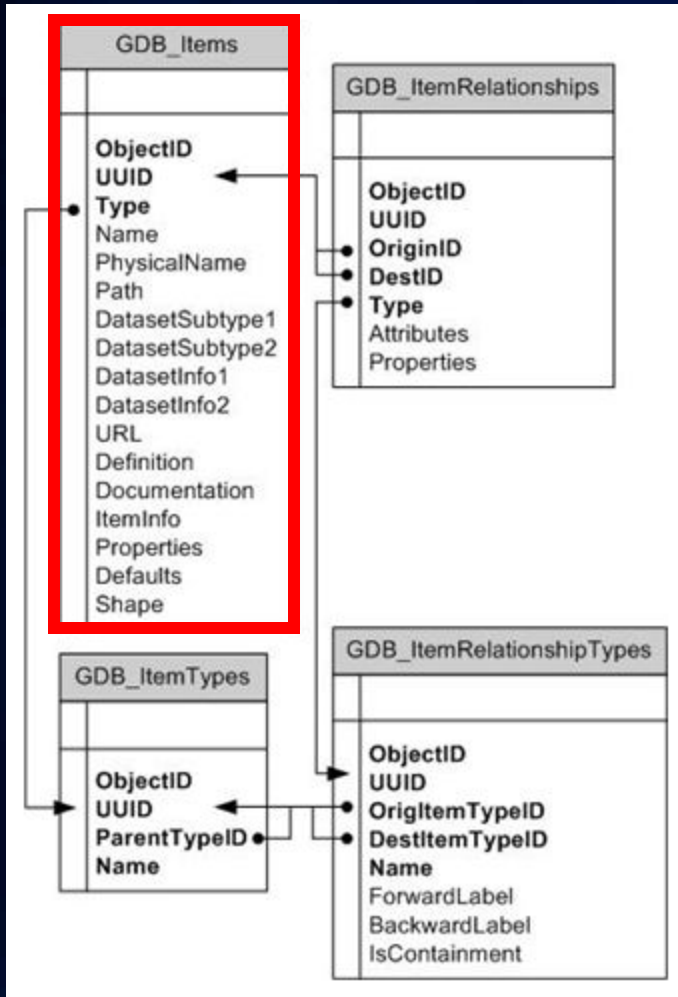
- System tables store definitions, rules, and behavior for datasets
- Tracks contents within a geodatabase
- 4 primary tables
- Geodatabase schema is stored within an XML field



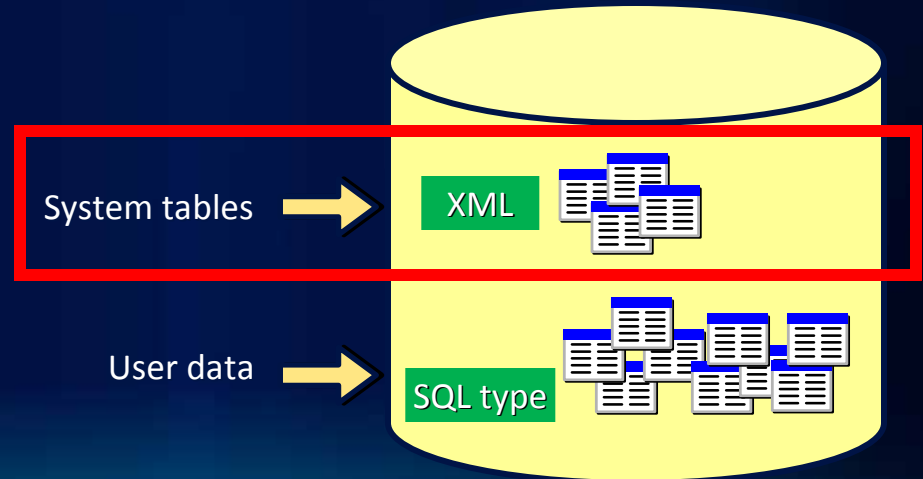
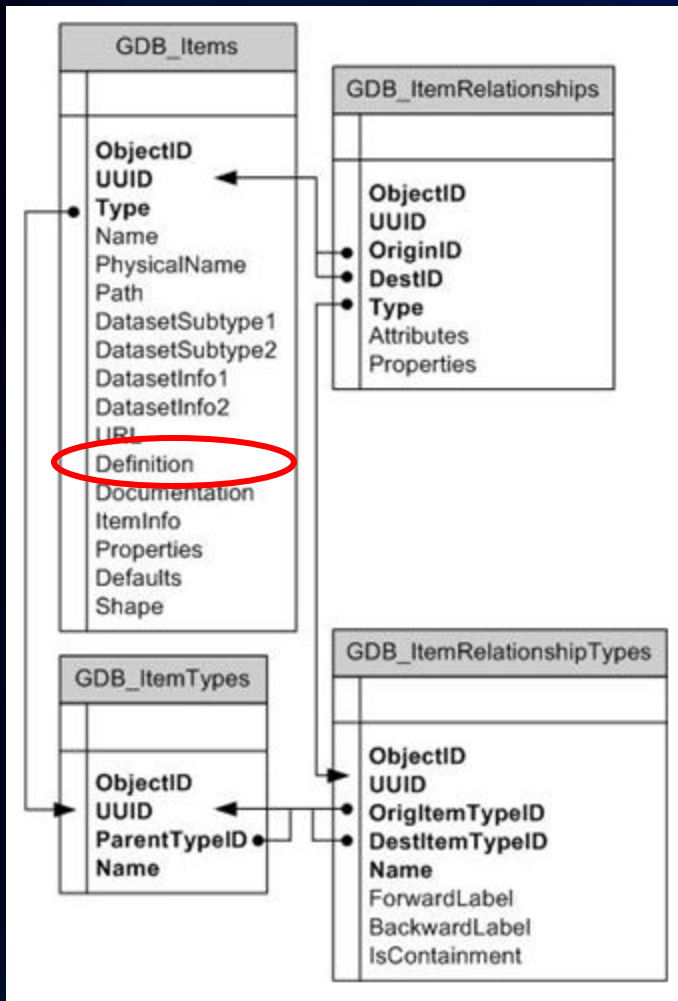
Geodatabase Schema...



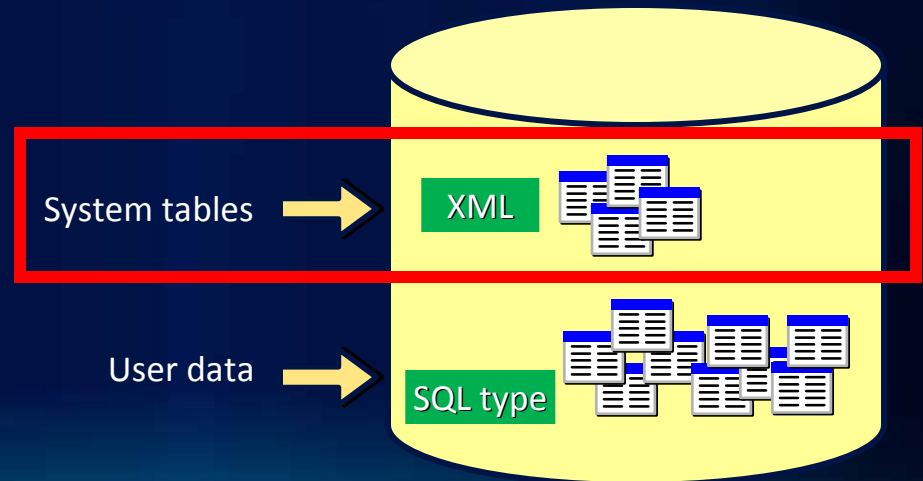
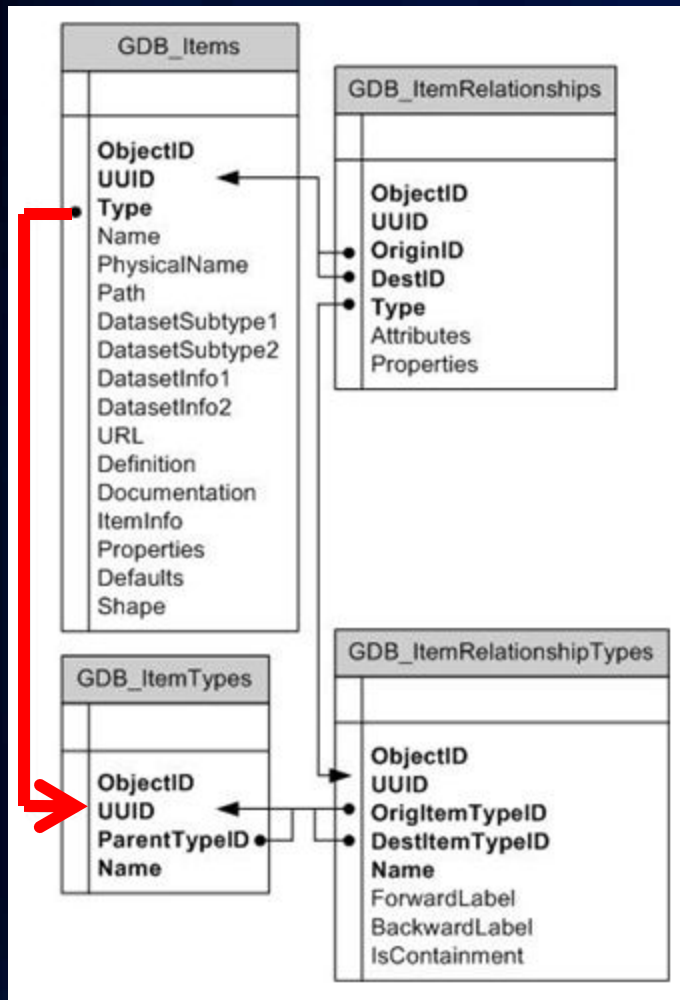
Geodatabase Schema...



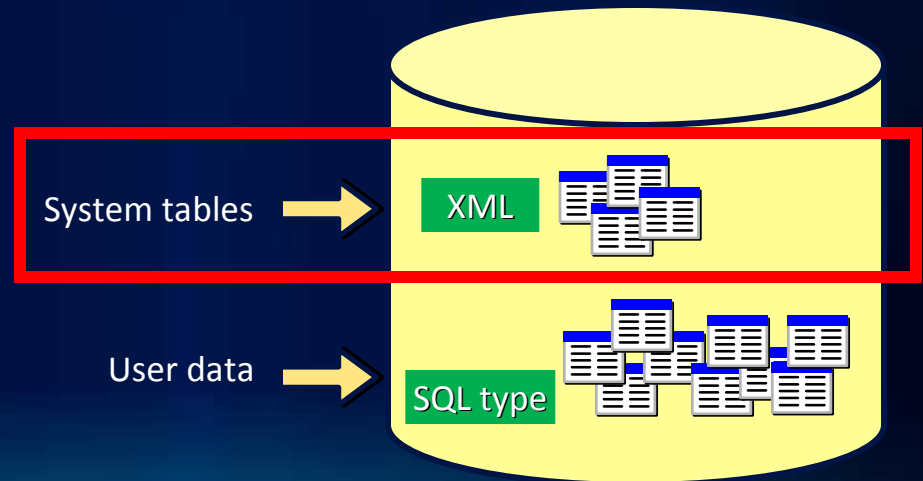
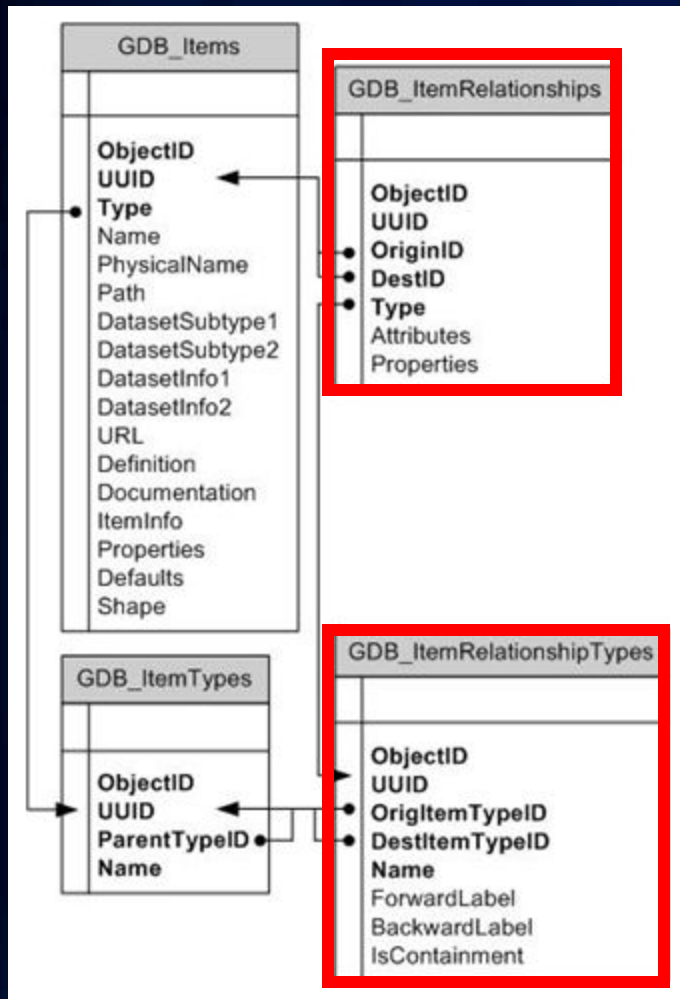
Geodatabase Schema...



Geodatabase Schema...



Geodatabase Schema...



Overview

- What is the Geodatabase?
- **Spatial Types and the Geodatabase**
- Accessing Geodatabase through SQL

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?

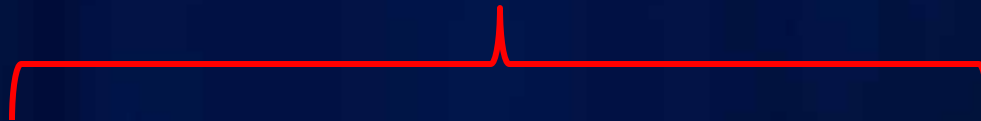
- **Using SQL with a spatial type you can**
 - **Create tables with a spatial attribute**
 - **Read and analyze the spatial data**
 - **Insert, update, and delete simple features**

What are the benefits of a spatial type?

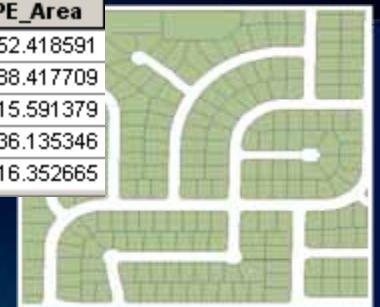
- Using SQL with a spatial type you can
 - Create tables with a spatial attribute
 - Read and analyze the spatial data
 - Insert, update, and delete simple geometry data

Spatial Type

SQL



OBJECTID *	SHAPE *	PROPERTY_ID *	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



Overview

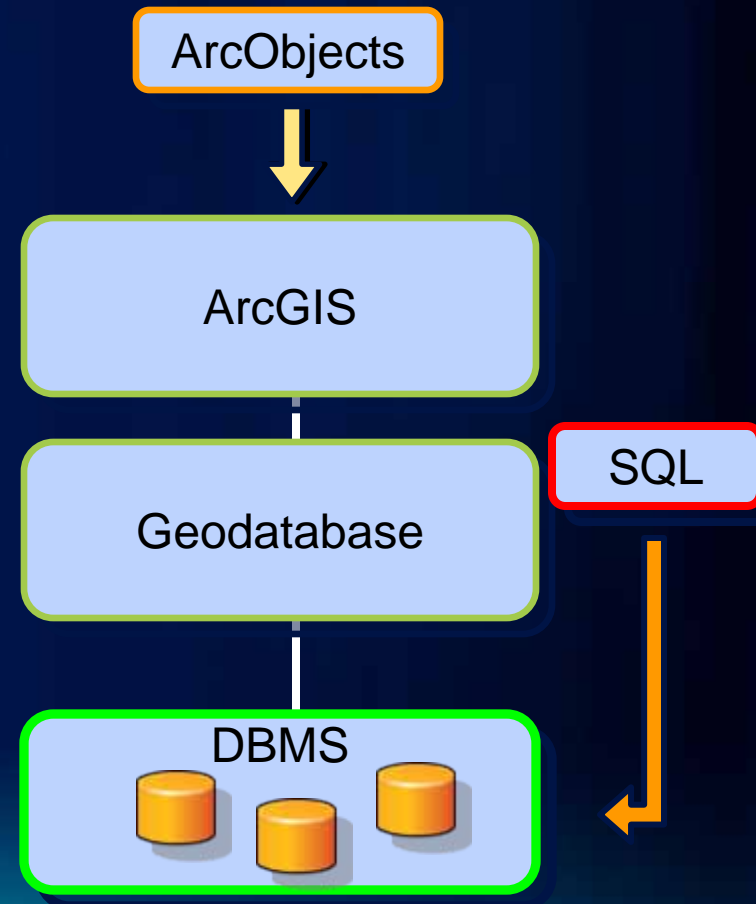
- **What is the Geodatabase?**
- **Spatial Types and the Geodatabase**
- **Accessing Geodatabase Schema through SQL**
 - **Querying Schema**
 - **Schema creation**
 - **Editing Tables/Feature Classes**
 - **Editing Versioned Tables/Feature Classes**

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**
 - 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 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

- **System tables overview**
- **Find datasets in feature dataset**
- **Find subtype field names and default codes**
- **Find subtype name/code pairs**
- **Get Range Domain Min/Max values**
- **Get Coded Value Domain code/value pairs**
- **Resolve table codes to CV Domain values**
- **Find all classes using a domain**

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 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 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

Adding a Feature

Inserts a row in the Adds table

6		
1	2	
3	4	5

Base Table

ObjectID	Perimeter	Bldg_Code
1	10105.15	02
2	10105.15	02
3	11348.31	02
4	10827.18	02
5	11348.31	02

Adds Table

ObjectID	Perimeter	Bldg_Code	SDE_State_ID
6	10105.15	02	27505

Deletes Table

Deleted_At	Deletes_Row_ID	SDE_State_ID

Editing versioned tables/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**

Accessing Geodatabase through SQL

- **Spatial and Attribute queries**
- **Editing tables/feature classes**
- **Editing versioned tables/feature classes demo**
- **Consuming SQL results with Query Layers**

Summary

- **Geodatabase built on 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**

Summary

- **Simplified Geodatabase Structure at 10.0**
 - Facilitates access to geodatabase information
- **Accessing Geodatabase Schema through SQL**
 - Many tasks possible now, some require ArcGIS
 - Provide ability to get at all content in the Geodatabase through SQL without ArcGIS/ArcObjects