Geodatabase Programming with SQL

Craig Gillgrass
Assumptions

- **Basic knowledge of SQL and relational databases**
- **Basic knowledge of the Geodatabase**
- **We’ll hold all questions till end**

Please turn off cell phones
Roadmap

• Using databases in ArcGIS

• Building on databases with the geodatabase

• Accessing the schema of a geodatabase through SQL

• Editing a geodatabase through SQL
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

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>SHAPE</th>
<th>PROPERTY_I</th>
<th>Res</th>
<th>Zoning_simple</th>
<th>SHAPE_Length</th>
<th>SHAPE_Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Polygon</td>
<td>5001</td>
<td>Non-Residential</td>
<td>&lt;Null&gt;</td>
<td>3597.780813</td>
<td>112552.418591</td>
</tr>
<tr>
<td>2</td>
<td>Polygon</td>
<td>5002</td>
<td>Non-Residential</td>
<td>&lt;Null&gt;</td>
<td>814.855837</td>
<td>18488.417709</td>
</tr>
<tr>
<td>3</td>
<td>Polygon</td>
<td>1003</td>
<td>Residential</td>
<td>Residential</td>
<td>489.655523</td>
<td>12815.591379</td>
</tr>
<tr>
<td>4</td>
<td>Polygon</td>
<td>1004</td>
<td>Residential</td>
<td>Residential</td>
<td>521.761248</td>
<td>14036.135346</td>
</tr>
<tr>
<td>5</td>
<td>Polygon</td>
<td>1005</td>
<td>Residential</td>
<td>Residential</td>
<td>453.479649</td>
<td>9816.352665</td>
</tr>
</tbody>
</table>
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

```sql
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

```sql
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

```sql
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

```sql
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

```sql
FROM dbo.HurricaneTracks_2005 hurricane, dbo.counties county, dbo.states states
WHERE hurricane.NAME = 'KATRINA' AND hurricane.Shape.STIntersects(county.shape) = 1
```
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
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

• 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…

- **GDB_Items**
  - ObjectId
  - UUID
  - Type
  - Name
  - PhysicalName
  - Path
  - DatasetSubtype1
  - DatasetSubtype2
  - DatasetInfo1
  - DatasetInfo2
  - URL
  - Definition
  - Documentation
  - ItemInfo
  - Properties
  - Defaults
  - Shape

- **GDB_ItemRelationships**
  - ObjectId
  - UUID
  - OriginID
  - DestID
  - Type
  - Attributes
  - Properties

- **GDB_ItemTypes**
  - ObjectId
  - UUID
  - ParentTypeID
  - Name

- **GDB_ItemRelationshipTypes**
  - ObjectId
  - UUID
  - OriginItemTypeID
  - DestItemTypeID
  - Name
  - ForwardLabel
  - BackwardLabel
  - IsContainment

**System tables**

**User data**
Geodatabase Schema...

- **System tables**: XML
- **User data**: SQL Type
### Geodatabase Schema

**System tables**
- **XML**

**User data**
- **SQL Type**

<table>
<thead>
<tr>
<th>Table</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDB_Items</td>
<td>ObjectID, UUID, Type, Name, PhysicalName, Path, DatasetSubtype1, DatasetSubtype2, DatasetInfo1, DatasetInfo2, URL, Definition, Documentation, ItemInfo, Properties, Defaults, Shape</td>
</tr>
<tr>
<td>GDB_ItemRelationships</td>
<td>ObjectID, UUID, OriginID, DestID, Type, Attributes, Properties</td>
</tr>
<tr>
<td>GDB_ItemTypes</td>
<td>ObjectID, UUID, ParentTypeID, Name, ForwardLabel, BackwardLabel, IsContainment</td>
</tr>
<tr>
<td>GDB_ItemRelationshipTypes</td>
<td>ObjectID, UUID, OrigItemTypeID, DestItemTypeID, Name, ForwardLabel, BackwardLabel, IsContainment</td>
</tr>
</tbody>
</table>
Geodatabase Schema...

System tables
XML

User data
SQL Type
Geodatabase Schema...

System tables: XML

User data: SQL Type
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…
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
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

```sql
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

- 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

<table>
<thead>
<tr>
<th>Base Table</th>
<th>Adds Table</th>
<th>Deletes Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObjectID</strong></td>
<td><strong>ObjectID</strong></td>
<td><strong>Deleted_At</strong></td>
</tr>
<tr>
<td><strong>Perimeter</strong></td>
<td><strong>Perimeter</strong></td>
<td><strong>Deletes_Row_ID</strong></td>
</tr>
<tr>
<td><strong>Bldg_Code</strong></td>
<td><strong>Bldg_Code</strong></td>
<td><strong>SDE_State_ID</strong></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10105.15</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10105.15</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>11348.31</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10827.18</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>11348.31</td>
</tr>
</tbody>
</table>
Editing versioned tables and feature classes

• Use versioned views
  - Created when data registered as versioned with ArcGIS 10.1 and later
  - Conditions where no view is present; use the “Enable SQL Access” command.

• Must use several stored procedures/commands installed with the geodatabase
  - 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 a geodatabase through SQL

- Editing
  - Versioned and Non Versioned Classes
  - Working with Views
What’s new in ArcGIS 10.3?

- Nine tools added to the Geodatabase Administration toolset for enterprise geodatabase management
  - Replace functionality previously performed using ArcSDE administration command line utilities.
- Configure Geodatabase Log File Tables
- Create Raster Type
- Delete Schema Geodatabase
- Diagnose Version Metadata
- Diagnose Version Tables
- Export Geodatabase Configuration Keyword
- Import Geodatabase Configuration Keyword
- Repair Version Metadata
- Repair Version Tables
Summary

• **GDB is open** to SQL Devs

• Through SQL use XML field in the GDB_Items table

• Can also edit data through SQL