Working with Geodatabases Using SQL and Python

Jochen Manegold
Gerhard Trichtl

ESRI EUROPEAN DEVELOPER SUMMIT
Geodatabase – What is this?

• 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
Tables in Geodatabase - Geodatabase system tables

- System tables store definitions, rules, and behavior for datasets
- Tracks contents within a geodatabase
- Stores some database level metadata
  - Versions, domains, etc.
- Admin operations:
  - Version management
  - Connection management
  - Geodatabase upgrade
Tables in Geodatabase - User defined Tables

- Stores the content of each dataset in the geodatabase
  - Datasets are stored in one or more tables
- Administrative Operations:
  - Granting/revoking privileges
  - Updating statistics/indexes
  - Registering as versioned
  - Adding global id’s
  - Enabling editor tracking
You can access spatial or non-spatial data in a DBMS to use in ArcGIS

<table>
<thead>
<tr>
<th>Geodatabase</th>
<th>Database – Simple Feature Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>ALTIBASE (deprecated)</td>
</tr>
<tr>
<td>Informix</td>
<td>Dameng</td>
</tr>
<tr>
<td>ORACLE</td>
<td>Teradata</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Netezza (deprecated)</td>
</tr>
<tr>
<td>Microsoft SQLServer</td>
<td>DB2, Informix</td>
</tr>
<tr>
<td>SAP HANA</td>
<td>ORACLE, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQLServer, SAP HANA</td>
</tr>
</tbody>
</table>
Working with Geodatabases using Python or how you create your own Geodatabase

by Jochen Manegold
The Scenario

- I am the database administrator in our office
- I want to create a Geodatabase with featureclass and data
- Simon should be the data-owner – responsible for maintenance and data quality
- Clare and Richie are responsible to capture the data
- They want to edit the data in an isolated editor environment (Versioning)

- As this happens in many different ways, I want to automate the setup of this scenario
The Scenario – a database, users, a table, data and a version tree

- Database
- Users: simon, clare, richie
- Table
- Data
- Version tree:
  - sde.Default: protected
  - simon.simon: public
  - clare.clare: private
  - richie.richie: private
The Tools

• Database Server (f.e. PostgreSQL)
• ArcGIS Pro Geoprocessing Framework – Standard or Advanced
• Scripting Environment for Python (f.e. PyScripter)
The Steps

1. Create an Enterprise Geodatabase
2. Create an Administrator Connection to the Geodatabase
3. Create the Database Users (Simon, Clare, Richie)
4. Create a Database Role (pg_giseditor) and add the Users to that Role
5. Create User Connections to the Geodatabase for each User
6. Create a Featureclass for the data, add fields and indexes
7. Load Data to the Featureclass
8. Register Featureclass 'as versioned'
9. Create a QA-Version for Simon
10. Create an Edit-Version for Clare and an Edit-Version for Richie
11. Create a User Connection to the Version for each User
12. Grant Read and Write Access to the Featureclass for the pg_giseditor Role
ArcPy is a Python site package that provides a useful and productive way to perform geographic data analysis, data conversion, data management, and map automation with Python.
Let’s rock...

Demo
Ressources

- Geoprocessing Tools (Geodatabase Administration Toolset)
- ArcPy Functions for Enterprise Geodatabase
- ArcPy Class for Enterprise Geodatabase
AcceptConnections

Zusammenfassung

Allows an administrator to enable or disable the ability of nonadministrative users to make connections to an enterprise geodatabase.

Auswertung

The AcceptConnections function is used by an administrative user to temporarily block connections to an Enterprise geodatabase. This function is used to complement the Connections tab on an Enterprise geodatabase properties page found in ArcGIS Desktop.

- The AcceptConnections function must utilize an administrative connection to the database.
- If this function is attempted to be run by a nonadministrative user the function will fail.

Syntax

AcceptConnections (sde_workspace, accept_connections)
ArcPy Class for Enterprise Geodatabase

- `arcpy.ArcSDESQLExecute`

<table>
<thead>
<tr>
<th>Methode</th>
<th>Erklärung</th>
</tr>
</thead>
<tbody>
<tr>
<td>commitTransaction ()</td>
<td>No DML statements will be committed until the CommitTransaction method is called.</td>
</tr>
<tr>
<td></td>
<td>🔄 Hinweis: A commit may also occur when the connection to ArcSDE is terminated (check specific DBMS documentation to see how each DBMS deals with a disconnect while in a transaction).</td>
</tr>
<tr>
<td>execute (sql_statement)</td>
<td>Sends the SQL statement to the database via an ArcSDE connection. If execute is run outside of a transaction, a commit will automatically take place once the SQL DML (INSERT, UPDATE, DELETE...) statement has been executed.</td>
</tr>
<tr>
<td>rollbackTransaction ()</td>
<td>Rollback any DML operations to the previous commit.</td>
</tr>
<tr>
<td>startTransaction ()</td>
<td>To control when your changes are committed to the database, call the startTransaction method before calling execute. This starts a transaction and no DML statements will be committed until the commitTransaction method is called.</td>
</tr>
</tbody>
</table>

[https://pro.arcgis.com/de/pro-app/arcpy/classes/arcsdesqlexecute.htm](https://pro.arcgis.com/de/pro-app/arcpy/classes/arcsdesqlexecute.htm)
Let’s rock again...

Demo
Access with SQL
Gerhard Trichtl
Accessing your geodatabase using 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
  - Know your data
  - Use discovery functions
Information from Geodatabase System Tables
Geodatabase schema – four main system tables

- **GDB_Items**
  - List all geodatabase items

- **GDB_ItemTypes**
  - Fixed list of items

- **GDB_ItemRelationships**
  - List all relationships

- **GDB_ItemRelationshipTypes**
  - Fixed list of relationships

- **XML document for each item**

- **Native XML**
  - SQLServer
  - PostgreSQL
  - DB2

- **ArcSDE XML**
  - ORACLE
  - Informix
List of Domains in Geodatabase

- See Example:

```
-- Get List of Domains and Owners
SELECT items.Name AS "Domain Name",
       items.Definition.value('(/*/Owner)[1]', 'nvarchar(max)') AS "Owner"
FROM sde.GDB_ITEMS AS items INNER JOIN sde.GDB_ITEMTYPES AS itemtypes
ON items.Type = itemtypes.UUID
WHERE itemtypes.Name IN ('Coded Value Domain', 'Range Domain')
```
Get List of Featureclasses with a specific Domain

- See Example:
Function would be also available within ArcGIS Pro 2.5 - DomainUsage
Geodatabase-Version/Versioned Featureclasses

- Geodatabase Version:

- Versioned Featureclasses:
Additional Repository Tables

- Beside GDB_xxx-Tables there exist other Tables to Maintain Enterprise Geodatabase.
- Overview of the models in the Desktop-Installation-Folder\Documentation.
Information/Analysis from User-Tables
Quering geodatabase user-data

- Why use SQL when I have a GIS?
  - Use power of DBMS enginge to query and analyze your data
  - DBMS spatial methods for performing spatial analysis
  - Bridge between GIS and Business Intelligence / Insights
  - Sometimes you want a single result and not a map
What is a Spatial Type

- A Type that stores geometry data in a single spatial attribute
  - Geometry type, coordinates, dimension, spatial reference

- Spatial index
  - Improves spatial search

- Relational and geometry operations and functions
  - Constructors – creates new geometry
  - Accessors – return property of a geometry
  - Relational – perform spatial operations
  - Geometry – transform from one geometry to another
Spatial Type
Benefits for a Spatial Type

• With SQL and a Spatial Type you can
  - Create Tables with a spatial attribute
  - Read and analyze spatial data
  - Insert, update and delete simple features

• Enhances efficiency
  - Data and methods are stored in the database
  - Applications access native dbms type

• Access using common API’s and SQL
  - Standard functions
  - Well-known interchange formats
Viewing database data in ArcGIS

- SQL Query with QueryLayer
Create geodatabase feature classes using SQL

• Use SQL to create and populate tables

```
USE TESTDB
GO
CREATE TABLE SQLFeatureClass (OBJECTID INT,
NAME VARCHAR(20),
SHAPE GEOMETRY);
GO
```

• Need to register the table with the geodatabase to participate in the geodatabase functionality
Editing geodatabase feature classes using SQL

• What can you edit?
  - Simple features (points, lines, polygons)
  - Without geodatabase behavior
  - Use the Is_Simple function to determine whether your data can be edited

• Editing non-versioned tables
  - Edit tables directly

• Editing versioned tables
  - Edit special versioned views instead of tables
Populate fields with Spatial functions
Scenario for Showcase

• We want to add features via SQL by coordinates (eg. Addresses) from a 3rd Party Application

• We want to have information from a polygon featureclass in the created point (e.g. Name of Postal District)
prompt: Check SRID for TRIGGER:
SELECT SRID from SDE.ST_GEOMETRY_COLUMNS where TABLE_NAME = 'POINT_BSP3';

prompt: Create TRIGGERS:
CREATE OR REPLACE TRIGGER TRIGGER_POINT_BSP3 BEFORE INSERT ON POINT_BSP3 FOR EACH ROW
declare
amt varchar(100);

BEGIN

-- Got Geometry - important is to use the correct SRID (see statement on top)
:NEW.shape := sde.ST_PointFromText('POINT('||NEW.RECHTSWERT||' '||NEW. HOCHWERT||',300014);
-- Now Select based on Point Geometry the PostOffice
SELECT POSTMT
into amt
from plz_bsp3 where sde.st_intersects(sde.ST_PointFromText('POINT('||NEW.RECHTSWERT||' '||NEW. HOCHWERT||',300014), shape) = 1;
-- And Use value for the column on points
:NEW.POSTMT := amt;

END;
/
quit;
Code for Insert

```
-- Prepare - get REGISTRATION IDENTIFIED
SELECT registration_id FROM sde.table_registry WHERE table_name = 'POINT_BSP3' and owner = 'TEST2';

prompt Insert values in Database:
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_ddl.next_row_id('TEST2', 83800), 'Strandgasse 1', 435275.331335);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_ddl.next_row_id('TEST2', 83800), 'Pruggern Platz 4', 441763.335079);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_ddl.next_row_id('TEST2', 83800), 'Schladminger Ried 5', 430494, 325517);
quit
```
Important how to maintain „OBJECTID“

- „OBJECTID“ is Maintained by Geodatabase, so „OBJECTID“ couldn’t be directly inserted with a value
- Value needs to be inserted with the function sde.version_user_dll.next_row_id, which get the next available „OBJECTID“

Wrong Insert

```sql
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (1,'Strandgasse 1',435275,331335);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (2,'Fuggerer Platz 4',441763,335079);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (3,'Schladminger Ried 5',430494,325517);
```

Obtain RegistrationID from TABLE_REGISTRY-Table

```sql
-- Prepare - get REGISTRATION IDENTIFIED
SELECT registration_id FROM sde.table_registry WHERE table_name = 'POINT_BSP3' and owner = 'TEST2';
```

Correct Insert with function

```sql
prompt Insert values in Database:
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_dll.next_row_id('TEST', 83900),'Strandgasse 1',435275,331335);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_dll.next_row_id('TEST', 83900),'Fuggerer Platz 4',441763,335079);
INSERT INTO POINT_BSP3 (OBJECTID,ADRESSE,RECHTSWERT,HÖCHWERT) values (sde.version_user_dll.next_row_id('TEST', 83900),'Schladminger Ried 5',430494,325517);
```
### ST_Funcions

Functions to work with the Geometry adn data

<table>
<thead>
<tr>
<th>ST_AsBinary</th>
<th>ST_LineFromShape</th>
<th>ST_MPolyFromShape</th>
<th>ST_PointFromText</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PostgreSQL only)</td>
<td>(PostgreSQL only)</td>
<td>(Oracle only)</td>
<td>(Oracle only)</td>
</tr>
<tr>
<td>ST AsText</td>
<td>ST_LineFromText</td>
<td>ST_MPolyFromText</td>
<td>ST_PointFromWKB</td>
</tr>
<tr>
<td>(Oracle only)</td>
<td>(Oracle only)</td>
<td>(Oracle only)</td>
<td></td>
</tr>
<tr>
<td>ST_Curve</td>
<td>ST_LineFromWKB</td>
<td>ST_MPolyFromWKB</td>
<td>ST_PolyFromShape</td>
</tr>
<tr>
<td>(Oracle only)</td>
<td>(PostgreSQL only)</td>
<td>(Oracle only)</td>
<td>(PostgreSQL only)</td>
</tr>
<tr>
<td>ST_GeomCollection</td>
<td>ST_LineString</td>
<td>ST_MultiCurve</td>
<td>ST_PolyFromText</td>
</tr>
<tr>
<td>ST_GeomCollFromShape</td>
<td>ST_MLineFromShape</td>
<td>ST_MultiLineString</td>
<td>ST_PolyFromWKB</td>
</tr>
<tr>
<td>(PostgreSQL only)</td>
<td>(PostgreSQL only)</td>
<td>(Oracle only)</td>
<td></td>
</tr>
<tr>
<td>ST_GeomCollFromWKB</td>
<td>ST_MLineFromText</td>
<td>ST_MultiPoint</td>
<td>ST_Polygon</td>
</tr>
<tr>
<td>(PostgreSQL only)</td>
<td>(Oracle only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST_Geometry</td>
<td>ST_MLineFromWKB</td>
<td>ST_MultiPolygon</td>
<td>ST_Surface</td>
</tr>
<tr>
<td>ST_GeomFromShape</td>
<td>ST_MPointFromShape</td>
<td>ST_MultiSurface</td>
<td>ST_Transform</td>
</tr>
<tr>
<td>(PostgreSQL only)</td>
<td>(PostgreSQL only)</td>
<td>(Oracle only)</td>
<td></td>
</tr>
<tr>
<td>ST_GeomFromText</td>
<td>ST_MPointFromText</td>
<td>ST_Point</td>
<td></td>
</tr>
<tr>
<td>(Oracle only)</td>
<td>(Oracle only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST_GeomFromWKB</td>
<td>ST_MPointFromWKB</td>
<td>ST_PointFromShape</td>
<td></td>
</tr>
<tr>
<td>(PostgreSQL only)</td>
<td>(PostgreSQL only)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Guidelines for using SQL and the geodatabase

- Understanding the geodatabase system and their structure

- Avoid changing data that affects geodatabase software behavior

- Geodatabase awareness
  - You have it
  - The database does not
**Guidelines for using SQL and the geodatabase**

<table>
<thead>
<tr>
<th></th>
<th>GDB System tables</th>
<th>Simple FC / Tables</th>
<th>Complex FC / Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QUERY</strong></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Edit/Update</strong></td>
<td>![Caution symbol]</td>
<td>✓</td>
<td>![Caution symbol]</td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td>![Caution symbol]</td>
<td>✓</td>
<td>![Caution symbol]</td>
</tr>
</tbody>
</table>
Guidlines for using SQL and the geodatabase

- **DO NOT** update the OBJECTID(row_id) value

- **DO NOT** modify geometries for feature classes participate in non simple data as
  - Topologies, geometric networks, network datasets, terrains, parcel fabric,
  - Geodatabase replication, schematic datasets, feature-linked annotation, …

- **DO NOT** update attributes that define geodatabase behavior
  - Enable/Disable attributes, ancillary attributes, weight attributes, …

- Use Is_Simple to check
Guidlines for using SQL and the geodatabase

• Do perform spatial operations

• Do query spatial and attribute information

• Do INSERT, UPDATE and DELETE geometries
  As long you pay attention to behavior

• Do INSERT, UPDATE and DELETE attribute data
  As long you pay attention to behavior

• Do write efficient SQL
Resources

• Comprehensive documentation covering
  - Accessing dataset properties
  - Editing geodatabase data
  - Esri spatial and raster type reference

• Get started at
Questions?
Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the feedback section

Complete answers and select “Submit”