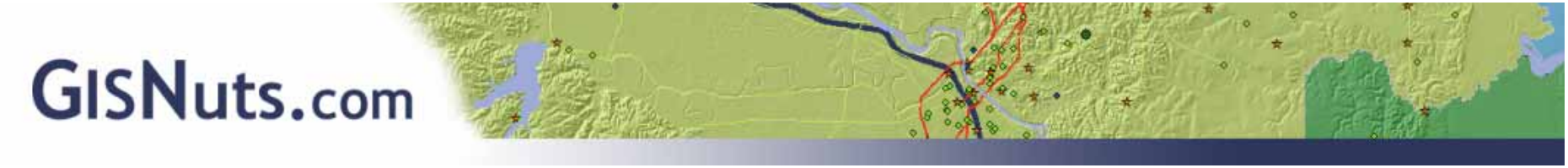


Using SQLite in ArcGIS Python Scripts

Leslie H. Morgan, Ph.D.
Leslie.Morgan@GISNuts.com

What is SQLite?

The most widely deployed database engine worldwide.

Transactional SQL database.

Implements most of SQL92.

Requires no configuration.

Small. Fast. Reliable.
Choose any three.
http://www.sqlite.org/

Open source and in the public domain.

It's Free!!!

SQLite Features

Simple and easy to use

Small Footprint

Fast

Supports terabyte-sized databases

Self-contained and cross-platform

ACID Transactions (Atomic, Consistent, Isolated, and Durable)

How do you access SQLite?

- The **sqlite3** module is a C library that provides a SQL interface compliant with the DB-API 2.0 specification.
- Written by Gerhard Häring
- Part of the Python Standard Library (version 2.5+)

```
>>> help("modules")

Please wait a moment while I gather a list of all available modules...

BaseHTTPServer      _sha512             imaplib             statvfs
Bastion             _socket             imghdr              string
BatchDefineSR       _sqlite3            imp                 stringold
BatchProject        _sre                importe00           stringprep
BatchPyramids       _ssl                imputil             strop
```

```
_json               hashlib             socket              xml
_locale             heapq               sqlite3             xmllib
_lsprof             hmac                sre                 xmlrpclib
_md5                hotshot             sre_compile         xxsubtype
```

Where is the data stored?

- In a single *.db file

```
>>> import sqlite3
>>> conn = sqlite3.connect(r"C:\Example.db")
>>> c = conn.cursor()
>>> c.execute('''CREATE TABLE states (name text, abbr text, pop integer)''')
```

- In memory

```
>>> import sqlite3
>>> conn = sqlite3.connect(":memory:")
>>> c = conn.cursor()
>>> c.execute('''CREATE TABLE states (name text, abbr text, pop integer)''')
```

What can you do?

- Write standard ANSI SQL

- Implements most of SQL92 (A few limitations)

- **SELECT DISTINCT** CLASS **FROM** GNIS_LA_2013

A Basic Example

Basic Example

```
import sqlite3
conn = sqlite3.connect(r"C:\GISNuts\Example.db")
c = conn.cursor()
c.execute('''SELECT DISTINCT CLASS FROM
                    GNIS_LA_2013''')
lstunique = c.fetchall()
conn.close()
```

SQL Examples

Example 2:

SELECT CLASS, Count(FEATURE_ID) **FROM** GNIS_LA_2013

GROUP BY CLASS

ORDER BY Count(FEATURE_ID) **DESC**

Example 3:

SELECT FEATURE_CLASS, Count(FEATURE_ID) **FROM** GNIS_LA_2013

GROUP BY FEATURE_CLASS

ORDER BY Count(FEATURE_ID) **DESC**

LIMIT 10

Code

```
import sqlite3
conn = sqlite3.connect(r"C:\GISNuts\Example.db")
c = conn.cursor()
c.execute('''SELECT CLASS, Count(FEATURE_ID) FROM GNIS_LA_2013
                    GROUP BY CLASS
                    ORDER BY Count(FEATURE_ID) DESC''')
lstuniquecount = c.fetchall()
c.execute('''SELECT CLASS, Count(FEATURE_ID) FROM GNIS_LA_2013
                    GROUP BY CLASS
                    ORDER BY Count(FEATURE_ID) DESC
                    LIMIT 10''')
lstuniquecount10 = c.fetchall()
conn.close()
```

SQL Example

tbl1

ID	FirstName	LastName	Address	ZipCode
1	John	Smith	123 Main Street	70459
2	Jane	Doe	1015 2nd Street	84256
3	Tina	Smart	456 Hwy 1	47825
4	Robert	Powers	634 Bolton Avenue	65278
5	Stephanie	Carlin	5683 Twin Bridges R	25486

tbl2

ID	FirstName	LastName	Address	ZipCode
3	Tina	Smart	456 Hwy 1	47825
4	Robert	Powers	634 Bolton Avenue	65278
5	John	Smith	123 Main Street	70459
6	Jane	Doe	1015 2nd Street	84246
7	Mark	Brown	432 1st Avenue	76489

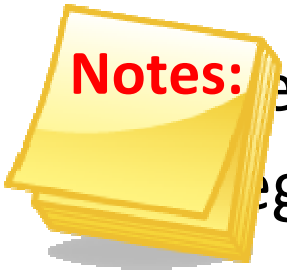
```
SELECT tbl1.ID, tbl2.ID
FROM tbl1, tbl2
WHERE ((tbl1.Address = tbl2.Address) AND (tbl1.LastName
= tbl2.LastName) AND (tbl1.FirstName = tbl2.FirstName)
AND (tbl1.ZipCode<>tbl2.ZipCode))
```

Code

```
import sqlite3

conn = sqlite3.connect(r"C:\GISNuts\Example.db")

c = conn.cursor()

c.execute('''SELECT tbl1.ID1, tbl2.ID2 FROM tbl1, tbl2
                WHERE ((tbl1.Address = tbl2.Address)
                AND (tbl1.LastName = tbl2.LastName)
                AND (tbl1.FirstName = tbl2.FirstName)
                AND (tbl1.ZipCode<>tbl2.ZipCode))''')

lstdiffzip = c.fetchall()

conn.close()
```

How to Get Data from ArcGIS

TableToNumPyArray (in_table, field_names, {where_clause}, {skip_nulls}, {null_value})



e, raster, and BLOB fields are not supported

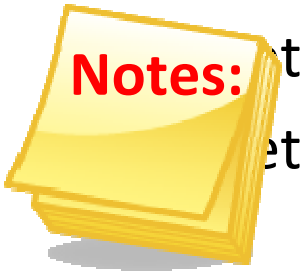
eger fields cannot contain nulls.

- rows can be skipped
- value can be assigned to null values

3)Pull only the fields you need and delete arrays when finished with them to avoid memory errors

How to Get Data from ArcGIS

FeatureClassToNumPyArray (in_table, field_names, {where_clause}, spatial_reference, {explode_to_points}, {skip_nulls}, {null_value})



- try fields are also not supported.
- try properties can be added to the array using tokens:

SHAPE@XY	The feature's centroid x,y coordinates
SHAPE@TRUECENTROID	The feature's true centroid x,y coordinates
SHAPE@X	The feature's x-coordinate
SHAPE@Y	The feature's y-coordinate
SHAPE@Z	The feature's z-coordinate
SHAPE@M	The feature's m-value
SHAPE@AREA	The feature's area
SHAPE@LENGTH	The feature's length

Example

```
import arcpy, sqlite3
conn = sqlite3.connect(r":memory:")
c = conn.cursor()
tbl = r"C:\GISNuts\GISNuts.gdb\GNIS_LA_2013"
array = arcpy.da.TableToNumPyArray(tbl, ["CLASS"])
c.execute("Create table GNIS (Class text)")
c.executemany("Insert into GNIS values (?)", array)
conn.commit()
del array
c.execute("SELECT DISTINCT Class from GNIS")
lstunique = c.fetchall()
conn.close()
```

Example

```
import arcpy, sqlite3
conn = sqlite3.connect(r":memory:")
c = conn.cursor()
fc = r"C:\GISNuts\GISNuts.gdb\GNIS_LA_2013"
array = arcpy.da.FeatureClassToNumPyArray(fc, ["SHAPE@X",
        "SHAPE@Y"], '"PRIM_LAT_DEC" <>  0')
c.execute("Create table GNISCoords (X real, Y real)")
c.executemany("Insert into GNISCoords values (?, ?)", array)
conn.commit()
del array
c.execute("SELECT Max(X), Min(X), Max(Y), Min(Y) from
        GNISCoords")
bndbox = c.fetchall()
conn.close()
```

What About Dates?

- NumPy arrays can not store dates
- SQLite does not have a Date field type
- Dates and times are stored as TEXT, REAL, or INTEGER values in SQLite:
  - **TEXT** as ISO8601 strings ("YYYY-MM-DD HH:MM:SS.SSS")
  - **REAL** as Julian day numbers, the number of days since noon in Greenwich on November 24, 4714 B.C. according to the proleptic Gregorian calendar.
  - **INTEGER** as Unix Time, the number of seconds since 1970-01-01 00:00:00 UTC

Date Example

```
import arcpy, sqlite3

conn = sqlite3.connect(r":memory:")
c = conn.cursor()

fc = r"C:\GISNuts\GISNuts.gdb\GNIS_LA_2013"

c.execute("Create table GNISEdits (County text, FeatureID integer, DateCreated text, DateEdited
text)")

fields = ["COUNTY_NAME", "FEATURE_ID","DATE_CREATED", "DATE_EDITED"]
whereclause = '"DATE_CREATED" > date \'08/28/2005\' or "DATE_EDITED" > date \'08/28/2005\''

with arcpy.da.SearchCursor(fc, fields, whereclause) as cursor:
    for row in cursor:
        c.execute("Insert into GNISEdits values (?, ?, ?, ?)", row)
        conn.commit()

c.execute('''SELECT County, Count(FeatureID ), Max(DateCreated), Max(DateEdited) from GNISEdits
                GROUP BY County
                ORDER BY Count(FEATUREID) DESC;''')

editcount = c.fetchall()
conn.close()
```

Other Options

There's always another way . . . There's always a better way . . . If I had more time, I would have written less code. (Thomas Edison + Mark Twain)

```
SELECT CLASS, Count(FEATURE_ID) FROM GNIS_LA_2013
GROUP BY CLASS
ORDER BY Count(FEATURE_ID) DESC
```

Other Options for the Above Query:

1) **ArcGIS Statistics Tool**
2) **Manipulate Numpy Array**
3) **Search Cursor and Loop with Dictionary**

ArcGIS Statistics Tool

```
import arcpy

arcpy.env.overwriteOutput = 1

arcpy.env.workspace = r"C:\GISNuts\GISNuts.gdb"

fc = "GNIS_LA_2013"

outtbl = "ClassSummary"

arcpy.Statistics_analysis(fc,outtbl,"FEATURE_ID COUNT","CLASS")

uniquelistcountarry = arcpy.da.TableToNumPyArray(outtbl, ["CLASS",
"Frequency"])

uniquelistcountarry.sort(order = 'Frequency')

uniquelistcount = uniquelistcountarry.tolist()

uniquelistcount.reverse()
```

GISNuts.com

Manipulate Numpy Array

```
import arcpy

fc = r"C:\GISNuts\GISNuts.gdb\GNIS_LA_2013"

GNISArray = arcpy.da.TableToNumPyArray(fc, ["CLASS", "Feature_ID"])

uniqueclasses = list(set(GNISArray['CLASS']))

uniquelist = []

for value in uniqueclasses: uniquelist += [(value,
len(GNISArray[GNISArray['CLASS'] == value]))]

del GNISArray

uniquelist.sort(key=lambda x: x[1], reverse = True)
```

Search Cursor with Dictionary

```
import arcpy

fc = r"C:\GISNuts\GISNuts.gdb\GNIS_LA_2013"

classdict = {}
with arcpy.da.SearchCursor(fc,"Class") as cursor:
    for row in cursor:
        if classdict.has_key(row[0]):
            value = classdict[row[0]] + 1
            classdict[row[0]] = value
        else: classdict[row[0]] = 1

uniquelist = classdict.items()

uniquelist.sort(key=lambda x: x[1], reverse = True)
```

Thanks for Attending

Questions?

Leslie H. Morgan, Ph.D.

Leslie.Morgan@GISNuts.com

Twitter: @GISNuts

LinkedIn: Leslie Morgan

Presentation Link: www.GISNuts.com/SCAUG