31-database

April 29, 2016

1 SQL Databases

- easy to use from Python
- many different dbs are supported: sqlite3, mysql, postgres, oracle...

```
In [1]: # need some names to build a database, so made some random strings
        # but these are hard to read...

        import random

        def rs(n):
            # chr takes an ascii code and returns the letter in a string
            return(''.join( [ chr(random.randint(97, 122)) for j in range(n)] ))
        [ rs(4) for j in range(10)]
```

```
Out[1]: ['rice',
         'dttr',
         'hkax',
         'ndbz',
         'kmnt',
         'gclc',
         'jegm',
         'lsyx',
         'exrc',
         'txqs']
```

```
In [2]: # nltk corpus has 8,000 real names!
        # get some short ones

        import random
        import nltk
        def names(wcnt, wlen):
            names = [ w for w in nltk.corpus.names.words() if len(w) == wlen]
            # random doesn't have a 'random subset' routine
            # so do a shuffle
            random.shuffle(names)
            # then just take random names from the front
            return names[:wcnt]

        firsts = names(10, 4)
        lasts = names(10, 5)

        [firsts, lasts]
```

```
Out[2]: [['Edie',
          'Anet',
          'Kyle',
          'Kent',
          'Hiro',
          'Remy',
          'Emmy',
          'Etty',
          'Sena',
          'Andy'],
         ['Rosie',
          'Amata',
          'Conan',
          'Stace',
          'Stern',
          'Codee',
          'Candy',
          'Linet',
          'Alana',
          'Rahal']]
```

```
In [3]: # sqlite stores data in one file
        # delete old db if present

        import sqlite3
        import os
        import random
        dbf = '/tmp/3.db'
        os.remove(dbf)
```

```
In [4]: # make a connection and cursor

        con = sqlite3.connect(dbf)
        cur = con.cursor()
```

2 Create two tables

- grade(uni, course, grade)
- student(first, last, uni)

```
In [5]: cur.execute('create table grade (uni text, course text, grade real)')
```

```
Out[5]: <sqlite3.Cursor at 0x1105bbc00>
```

```
In [6]: cur.execute('create table student (first text, last text, uni text)')
```

```
Out[6]: <sqlite3.Cursor at 0x1105bbc00>
```

3 Build student table

```
In [7]: # first, last, uni

        students = [ [random.choice(firsts), random.choice(lasts)] for k in range(10)]
        students = [ [f, l, (f[0] + l[0] + str(random.randint(1000,9999))).lower()] for f,l in students
        students
```

```
Out[7]: [['Emmy', 'Conan', 'ec1753'],
         ['Hiro', 'Codee', 'hc3904'],
         ['Sena', 'Linet', 'sl5109'],
         ['Andy', 'Rosie', 'ar5224'],
         ['Edie', 'Codee', 'ec6284'],
         ['Remy', 'Stern', 'rs5796'],
         ['Sena', 'Candy', 'sc2792'],
         ['Hiro', 'Conan', 'hc2873'],
         ['Remy', 'Codee', 'rc6029'],
         ['Andy', 'Conan', 'ac4339']]
```

```
In [8]: # insert above list into db

        cur.executemany('insert into student values(?, ?, ?)', students)
```

```
Out[8]: <sqlite3.Cursor at 0x1105bbc00>
```

```
In [9]: # the select returns a generator

        sg = cur.execute('select uni, last from student')
        sg
```

```
Out[9]: <sqlite3.Cursor at 0x1105bbc00>
```

```
In [10]: rows = list(sg)
         rows
```

```
Out[10]: [('ec1753', 'Conan'),
          ('hc3904', 'Codee'),
          ('sl5109', 'Linet'),
          ('ar5224', 'Rosie'),
          ('ec6284', 'Codee'),
          ('rs5796', 'Stern'),
          ('sc2792', 'Candy'),
          ('hc2873', 'Conan'),
          ('rc6029', 'Codee'),
          ('ac4339', 'Conan')]
```

```
In [11]: # pull the uni out of the tuple

         unis = [t[0] for t in cur.execute('select uni from student')]
         unis
```

```
Out[11]: ['ec1753',
          'hc3904',
          'sl5109',
          'ar5224',
          'ec6284',
          'rs5796',
          'sc2792',
          'hc2873',
          'rc6029',
          'ac4339']
```

```
In [12]: # classes

         classes = ['COMSW {}'.format(random.randint(1000, 9999)) for j in range(5)]
         classes
```

```
Out[12]: ['COMSW 3902', 'COMSW 5346', 'COMSW 1204', 'COMSW 7139', 'COMSW 9959']

In [13]: # students take finals

         grades = [ [u , random.choice(classes), 70 + 30 * random.random() ] for u in unis ]
         grades

Out[13]: [['ec1753', 'COMSW 9959', 94.619147777136],
          ['hc3904', 'COMSW 9959', 80.81429428882099],
          ['sl5109', 'COMSW 1204', 79.7282376275868],
          ['ar5224', 'COMSW 3902', 73.26881451813999],
          ['ec6284', 'COMSW 7139', 87.02585843631144],
          ['rs5796', 'COMSW 5346', 89.56020416691507],
          ['sc2792', 'COMSW 9959', 94.05220083338104],
          ['hc2873', 'COMSW 9959', 86.27553337272258],
          ['rc6029', 'COMSW 9959', 81.44202789944794],
          ['ac4339', 'COMSW 1204', 94.98240774591221]]

In [14]: cur.executemany('insert into grade values(?, ?, ?)', grades)

Out[14]: <sqlite3.Cursor at 0x1105bbc00>

In [15]: # join the two tables on the uni field

         list(cur.execute('select first, last, student.uni, grade from student , grade  where student.u

Out[15]: [('Emmy', 'Conan', 'ec1753', 94.619147777136),
          ('Hiro', 'Codee', 'hc3904', 80.81429428882099),
          ('Sena', 'Linet', 'sl5109', 79.7282376275868),
          ('Andy', 'Rosie', 'ar5224', 73.26881451813999),
          ('Edie', 'Codee', 'ec6284', 87.02585843631144),
          ('Remy', 'Stern', 'rs5796', 89.56020416691507),
          ('Sena', 'Candy', 'sc2792', 94.05220083338104),
          ('Hiro', 'Conan', 'hc2873', 86.27553337272258),
          ('Remy', 'Codee', 'rc6029', 81.44202789944794),
          ('Andy', 'Conan', 'ac4339', 94.98240774591221)]

In [16]: # add a filter term

         q = 'select last, first,grade from student,grade'
         q += ' where grade>80 and student.uni = grade.uni'
         q += ' order by last'
         list(cur.execute(q))

Out[16]: [('Candy', 'Sena', 94.05220083338104),
          ('Codee', 'Hiro', 80.81429428882099),
          ('Codee', 'Edie', 87.02585843631144),
          ('Codee', 'Remy', 81.44202789944794),
          ('Conan', 'Emmy', 94.619147777136),
          ('Conan', 'Hiro', 86.27553337272258),
          ('Conan', 'Andy', 94.98240774591221),
          ('Stern', 'Remy', 89.56020416691507)]

In [17]: # always commit and close the connection

         con.commit()
         con.close()
```

```
In [18]: # data is persisted on disk, can read again
         # normally would use a with statement to automatically close

         res=None

         with sqlite3.connect(dbf) as con:
             cur = con.cursor()
             res = list(cur.execute(q))

         res
```

```
Out[18]: [('Candy', 'Sena', 94.05220083338104),
          ('Codee', 'Hiro', 80.81429428882099),
          ('Codee', 'Edie', 87.02585843631144),
          ('Codee', 'Remy', 81.44202789944794),
          ('Conan', 'Emmy', 94.619147777136),
          ('Conan', 'Hiro', 86.27553337272258),
          ('Conan', 'Andy', 94.98240774591221),
          ('Stern', 'Remy', 89.56020416691507)]
```

4 Object Relational Mappers

- maps objects into a relational database
- somewhat complex but very useful
- best known one is SQLAlchemy
- peewee is a simple one
- hibernate was a pioneering and hugely successful ORM for Java

5 NoSQL databases

- provide less functionality than SQL, but are more efficient and scale better
- mongodb is a popular one
- PyMongo is the python driver for mongodb