SQLite is an in-process library that implements a **self-contained, serverless, zero-configuration,** transactional SQL database engine. The code for SQLite is in the **public domain** and is thus free for use for any purpose, commercial or private.

(http://www.sqlite.org)

+ small, simple, fast

+ popular for being reliable

+ self-contained (primary key & foreign key)

+ application file is portable across all operating systems, 32-bit and 64-bit

+ Multiple processes for same application file (read and write without interference)

- limited in size to 140 terabytes

Create

```
import sqlite3

# create new db and make connection
conn = sqlite3.connect('zinc_db1.db')
c = conn.cursor()

# create table
c.execute('''CREATE TABLE zinc_db1
             (zinc_id TEXT, purchasable TEXT, non_rot_bonds INT)''')

# Insert one row of data
c.execute("INSERT INTO zinc_db1 VALUES ('ZINC00895032','YES', 4)")

# Insert multiple lines of data
multi_lines =[ ('ZINC00895033','YES', 1),
               ('ZINC00895034','NO', 0),
               ('ZINC00895035','YES', 3),
               ('ZINC00895036','YES', 9),
               ('ZINC00895037','YES', 10)
              ]
c.executemany('INSERT INTO zinc_db1 VALUES (?,?,?)', multi_lines)

# Save (commit) the changes
conn.commit()

# close connection
conn.close()
```

Query

```
import sqlite3

# open existing database
conn = sqlite3.connect('zinc_db1.db')
c = conn.cursor()

# print all lines ordered by number of non_rot_bonds
for row in c.execute('SELECT * FROM zinc_db1 ORDER BY non_rot_bonds'):
    print row

# print all lines that are purchasable and have <= 7 rotatable bonds
t = ('YES',7,)
for row in c.execute('SELECT * FROM zinc_db1 WHERE purchasable=? AND non_rot_bonds
<= ?', t):
    print row

# print all lines that are purchasable and have <= 7 rotatable bonds
t = ('YES',7,)
c.execute('SELECT * FROM zinc_db1 WHERE purchasable=? AND non_rot_bonds <= ?', t)
rows = c.fetchall()
for r in rows:
    print r

# close connection
conn.close()
```

Update

```
import sqlite3

# make connection to existing db
conn = sqlite3.connect('zinc_db1.db')
c = conn.cursor()

# update field
t = ('NO', 'ZINC00895033', )
c.execute("UPDATE zinc_db1 SET purchasable=? WHERE zinc_id=?", t)
print "Total number of rows changed:", conn.total_changes

# delete rows
t = ('NO', )
c.execute("DELETE FROM zinc_db1 WHERE purchasable=?", t)
print "Total number of rows deleted: ", conn.total_changes

# add column
c.execute("ALTER TABLE zinc_db1 ADD COLUMN 'keto_oxy' TEXT")

# save changes
conn.commit()

# print column names
c.execute("SELECT * FROM zinc_db1")
col_name_list = [tup[0] for tup in c.description]
print col_name_list

# close connection
conn.close()
```

CPU Time Benchmark

count_lines.py

```
start_time = time.clock()
lines = 0
    with open("drug_now.list", 'r') as fileobj:
        for line in fileobj:
            lines += 1
elapsed_time = time.clock() - start_time
    print "Time elapsed: {} seconds".format(elapsed_time)
    print "Read {} lines".format(lines)
```

CPU Time Benchmark

create_sqlite_db.py

```
start_time = time.clock()

conn = sqlite3.connect('zinc_db1.db')
c = conn.cursor()

c.execute('''CREATE TABLE zinc_db1
             (zinc_id TEXT, purchasable TEXT, non_rot_bonds INT)''')

lst = list()
with open("drug_now.list", "r") as myfile:
    for line in myfile:
        line = line.strip()
        lst.append((line, "Yes", None))
c.executemany("INSERT INTO zinc_db1 VALUES (?,?,?)", lst)
conn.commit()
conn.close()

elapsed_time = time.clock() - start_time
print "Time elapsed: {} seconds".format(elapsed_time)
```

CPU Time Benchmark

query_sqlite_db.py

```
start_time = time.clock()

conn = sqlite3.connect('zinc_db1.db')
c = conn.cursor()


lines = 0
t = ('YES',)
for row in c.execute('SELECT * FROM zinc_db1 WHERE purchasable=?', t):
    lines += 1

conn.close()

elapsed_time = time.clock() - start_time
print "Time elapsed: {} seconds”.format(elapsed_time)
print "Read {} lines".format(lines)
```

CPU Time Benchmark

Processing
6,059,029 entries

viento.bch.msu.edu

Red Hat Enterprise Linux Workstation

Release 6.4 (Santiago)

Kernel Linux 2.6.32-358.0.1.el6.x86_64

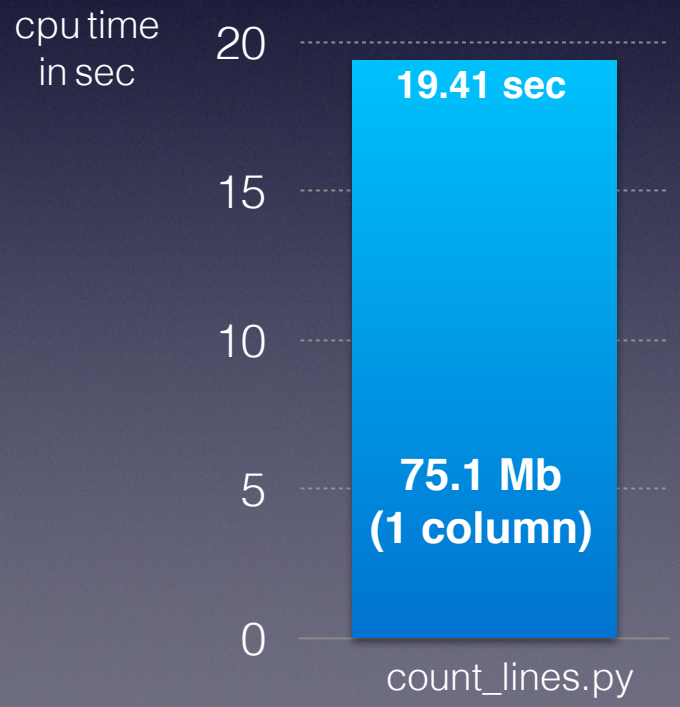
GNOME 2.28.2

Hardware

Memory:	5.6 GiB	
Processor 0:	Intel(R) Xeon(R) CPU	W3530 @ 2.80GHz
Processor 1:	Intel(R) Xeon(R) CPU	W3530 @ 2.80GHz
Processor 2:	Intel(R) Xeon(R) CPU	W3530 @ 2.80GHz
Processor 3:	Intel(R) Xeon(R) CPU	W3530 @ 2.80GHz

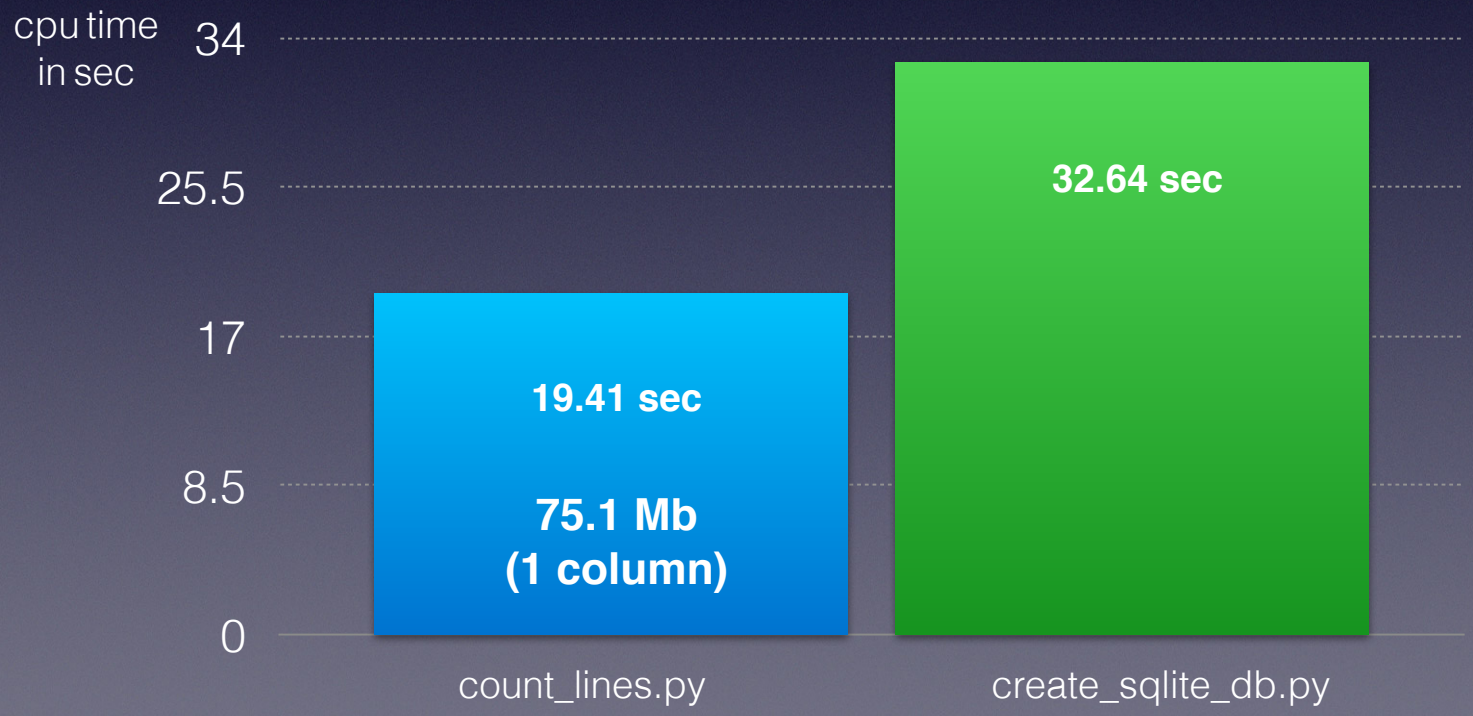
CPU Time Benchmark

Processing
6,059,029 entries



CPU Time Benchmark

Processing
6,059,029 entries



CPU Time Benchmark

Processing
6,059,029 entries