Be PROUD

Key-Value Storage Systems

… and beyond … with Python

Who the hell are you?

+

=

Ian Lewis
Company: BeProud
Tags: #python #django #redbull #mercurial
Twitter: IanMLewis
HP: http://www.ianlewis.org/

Why?

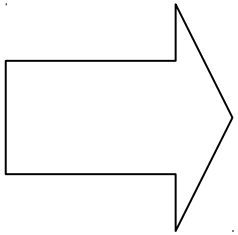
Pro:

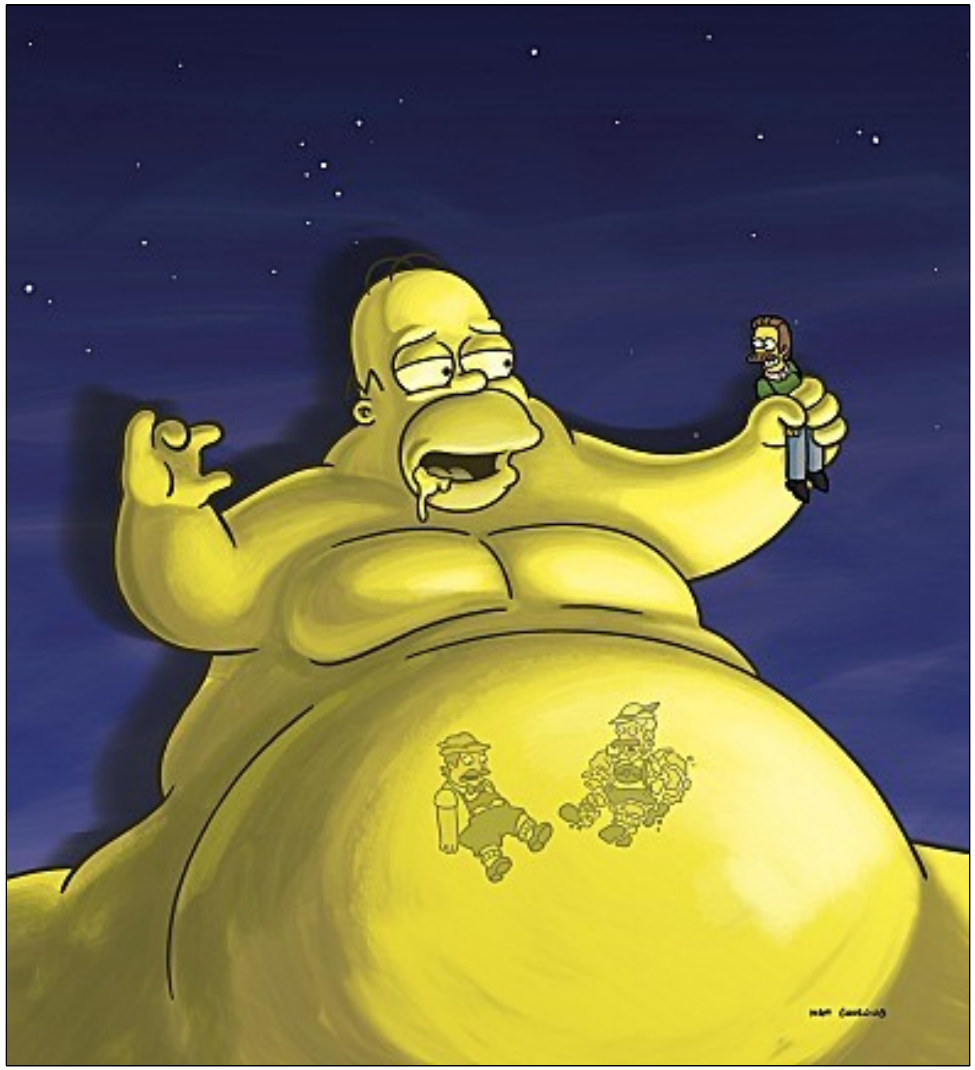
- Fast
- Simple

Con:

- Can't easily store complex relational data
- Can't do complex queries

What?





Memcached

Made by this guy →

- Not Persistent
- It's a cache so values go away
- Not useful as a database
- If you make websites and aren't using it you should be

Be PROUD

python-memcached

```
>>> import memcache
>>> client = memcache.Client(["127.0.0.1:11211"])
>>> client.set("test", 1)
True
>>> client.add("test", 1)
False
>>> client.add("test2", 1)
True
>>> client.set("test", 2)
True
>>> client.incr("test")
3
>>>
```

Tokyo-Cabinet/Tyrant

- Made by Mikio Hirabayashi for Mixi
- Persistent Key-Value Store
- Mmapped file
- Supports hash-table, b+trees, fixed-length array
- Replication

python-tokyotyrant/pytyrant

```
>>> import pytyrant
>>> t = pytyrant.PyTyrant.open('127.0.0.1', 1978)
>>> t['__test_key__'] = 'foo'
>>> t.concat('__test_key__', 'bar')
>>> print t['__test_key__']
foobar
>>> del t['__test_key__']
```

PY-

B+ Trees

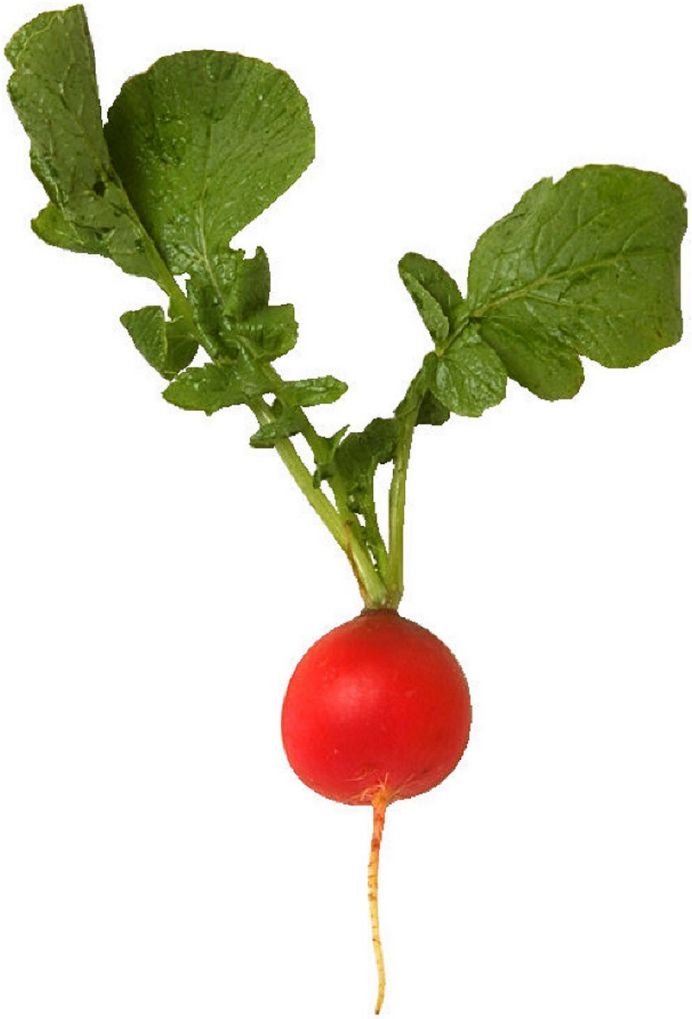
Be PROUD

pytc

```
>>> import pytc

>>> db = pytc.BDB('bdb.db', pytc.BDBOWRITER |
pytc.BDBOCREAT)
>>> db['niku'] = 'umai'
>>> db['niku']
'umai'
>>> db['ra-men'] = 'kuitai'
>>> db['ra-men']
'kuitai'
>>> for key in db:
>>>     print 'key:', key, ' value:', db[key]
'key: niki value: umai'
'key: ra-men value: kuitai'
```

Be PROUD

Redis =  ?

redis

Re-mote **Di**-ctionary **S**-erver

- Cool new KVS
- Data structures: Lists, Sets, Sorted Sets
- Queries: unions, intersection, complement (diff)
- In Memory (Persisted in background)
- Replication

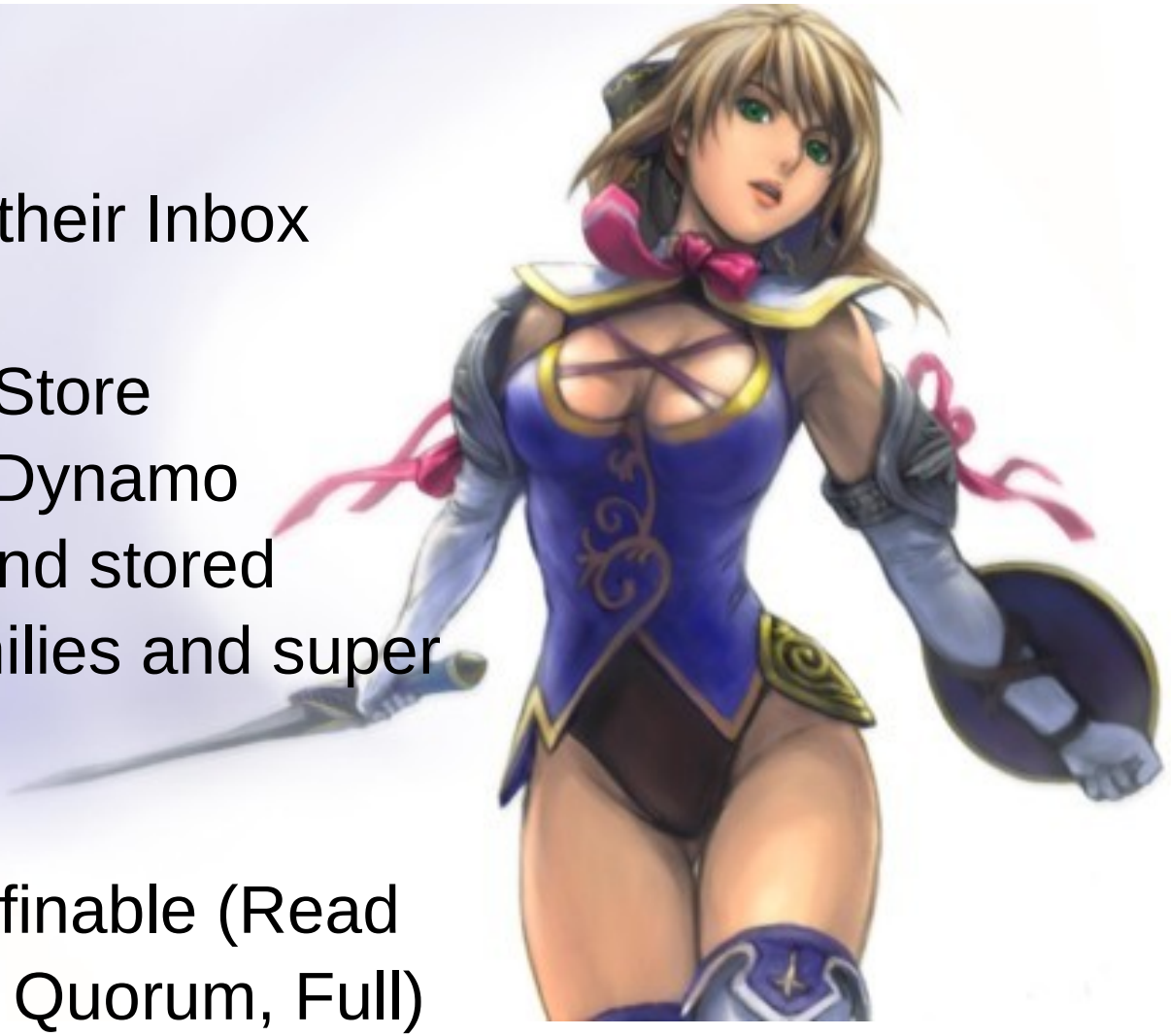
So I can't eat it?

redis-py

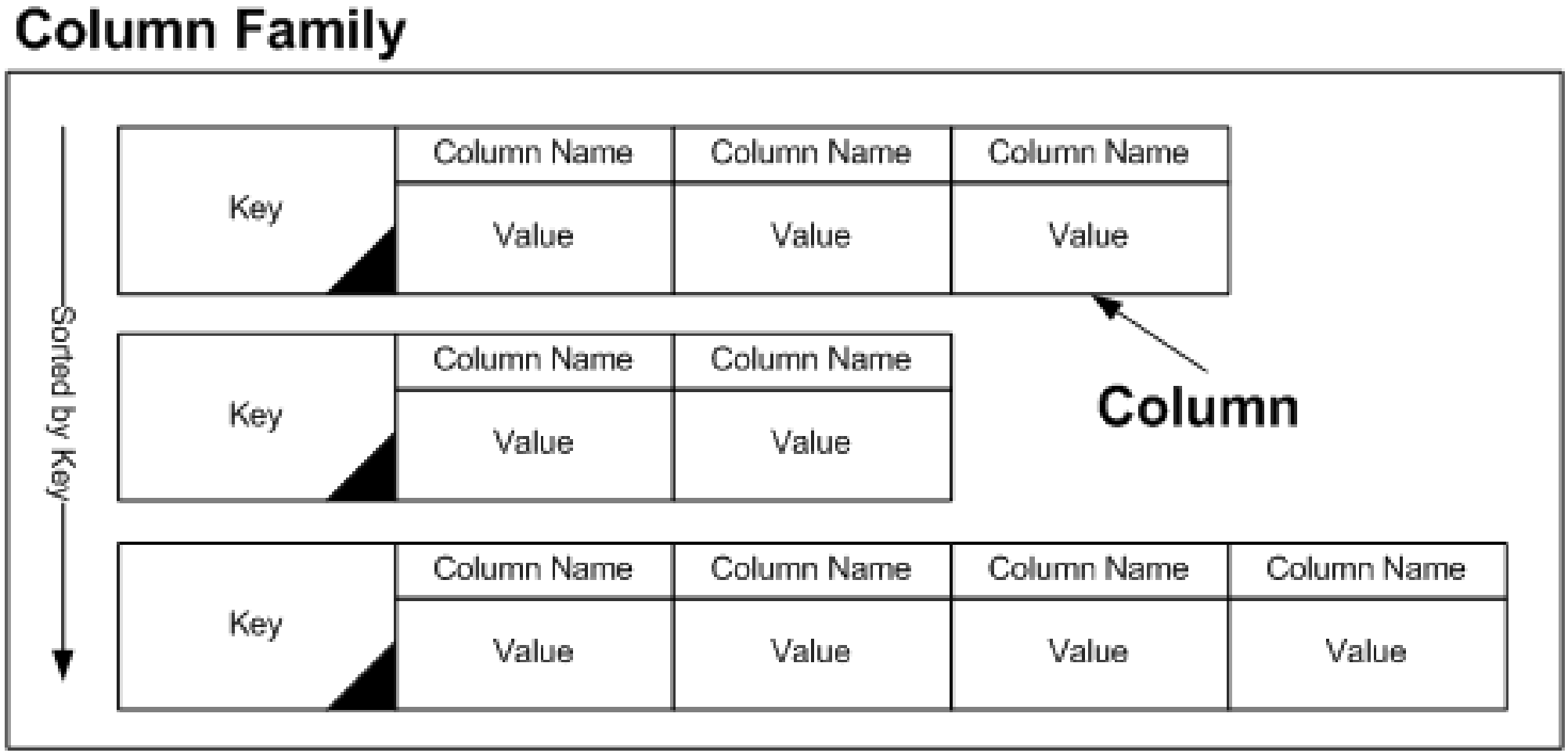
```
>>> from redis import Redis, ConnectionError,
  ResponseError
>>> r = Redis(db=9)
>>> r['a'] = 24.0
>>> r['a']
Decimal("24.0")
>>> r = Redis(db=9, float_fn=float)
>>> r['a']
24.0
>>> del r['a']
>>> print r.get('a') # r['a'] will raise KeyError
None
```

Cassandra

- Made at Facebook for their Inbox search
- Distributed Key-Value Store
- Influenced by Apache Dynamo
- Values are columnar and stored together in column families and super columns
- Eventually consistent
- Consistency is user definable (Read servers, Write servers, Quorum, Full)



Cassandra


Column Family
Key
Column Name
Column Name
Column Name
Value
Value
Value
Key
Column Name
Column Name
Value
Value
Column
Key
Column Name
Column Name
Column Name
Column Name
Value
Value
Value
Value
Sorted by Key

Be PROUD

pycassa

```
import pycassa
CLIENT = pycassa.connect_thread_local(framed_transport=True)

USER = pycassa.ColumnFamily(CLIENT, 'Twissandra', 'User',
    dict_class=OrderedDict)
...
TWEET = pycassa.ColumnFamily(CLIENT, 'Twissandra', 'Tweet',
    dict_class=OrderedDict)
TIMELINE = pycassa.ColumnFamily(CLIENT, 'Twissandra', 'Timeline',
    dict_class=OrderedDict)
...

timeline = TIMELINE.get(str(user_id), column_start=start,
column_count=limit,
            column_reversed=True)

tweets = TWEET.multiget(timeline.values())
```

Questions!