So we have some logs

How to get them into a readable format is beyond the scope of this presentation.

Let's assume you have your logs files in CSV format already:

In [1]:
```
from cStringIO import StringIO
log_file = StringIO(
    'posix_timestamp,elapsed,sys,user,queries,query_time,rows,'
        'accountid,userid,contactid,level,silo,method\n'
    '1343103150,0.062353,0,4,6,0.01690,3,'
        '12345,1,-1,3,invoice_InvoiceResource,search\n'
)
```

Then it's really easy to import the data into pandas:

In [2]:
```
import pandas as pd
from datetime import datetime
import gc

def posix_string_to_datetime(posix_string):
    return datetime.utcfromtimestamp(int(posix_string))

# Disabling GC for CPython dramatically improves the speed of bulk imports
# and won't actually use more memory unless we create circular garbage.
# For example, loading 1.8M records goes from taking 55s to taking 35s!
gc.disable()
df = pd.io.parsers.read_csv(
    log_file,
    # index_col is the first column, our posix_timestamp
    index_col=0,
    # Interpret the index column as a date
    parse_dates=0,
    date_parser=posix_string_to_datetime)
gc.enable()
```

In [3]:
```
df
```

Out[3]:

	elapsed	sys	user	queries	query_time	rows	accountid	use
posix_timestamp								
2012-07-24 04:12:30	0.062353	0	4	6	0.0169	3	12345	1

If you want to skip the minutes of waiting for your data to re-load (and feel ashamed about having to invoke `gc.disable()`), then get the `tables` (PyTables) module working so you can use HDF5!

In [4]:
```
store = pd.HDFStore('example.h5', complib='blosc')
store['example_data'] = df
```

In [5]:
```
store
```

Out[5]:
```
<class 'pandas.io.pytables.HDFStore'>
File path: example.h5
example_data     DataFrame
```

In [6]:
```
del store
```

In [7]:
```
store = pd.HDFStore('example.h5')
store['example_data']
```

Out[7]:

	elapsed	sys	user	queries	query_time	rows	accountid	use
posix_timestamp								
2012-07-24 04:12:30	0.062353	0	4	6	0.0169	3	12345	1

In [7]:
```

```