Extension of the HDFStore Documenation to show Table usage
Excellect original docs @ http://pandas.pydata.org/pandas-docs/stable/io.html#hdf5-pytables

HDFStore supports a Table object to enable HDF5 storage of appendable DataFrames and Panels

In [1]:
```
import pandas
import numpy as np
import os
```

In [2]:
```
if os.path.exists('store.h5'):
        os.remove('store.h5')
store = pandas.io.pytables.HDFStore('store.h5')
store
```

Out[2]:
```
<class 'pandas.io.pytables.HDFStore'>
File path: store.h5
Empty
```

In [3]:
```
p = pandas.Panel(np.random.randn(2, 30, 4), items=['Item1', 'Item2'],
    major_axis=pandas.date_range('1/1/2000', periods=30),
    minor_axis=['A', 'B', 'C', 'D'])
p
```

Out[3]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 30 (major) x 4 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-30 00:00:00
Minor axis: A to D
```

In [18]:
```
# regular store and retreive of a panel (.put is equivalent to store['mypa
store.put('mypanel',p)
mypanel = store.get('mypanel')
mypanel
```

Out[18]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 30 (major) x 4 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-30 00:00:00
Minor axis: A to D
```

In [19]:
```
# slice out 2 panels from the major_axis
p1 = p.ix[:,0:10,:]
p2 = p.ix[:,10:,:]
print p1
print p2
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 10 (major) x 4 (minor)
```

```
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-10 00:00:00
Minor axis: A to D
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 20 (major) x 4 (minor)
Items: Item1 to Item2
Major axis: 2000-01-11 00:00:00 to 2000-01-30 00:00:00
Minor axis: A to D
```

In [6]:
```
# store panels via append
store.append('appendpanel',p1)
store.append('appendpanel',p2)
store
```

Out[6]:
```
<class 'pandas.io.pytables.HDFStore'>
File path: store.h5
appendpanel      Panel (Table)
mypanel          Panel
```

In [7]:
```
# retrieve
appendpanel = store.select('appendpanel')
appendpanel
```

Out[7]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 30 (major) x 4 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-30 00:00:00
Minor axis: A to D
```

In [8]:
```
# let's select on the major_axis
import datetime
slicepanel = store.select('appendpanel',
    where = [ dict(field = 'index', op = '>=', value = datetime.datetime(2
slicepanel
```

Out[8]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 17 (major) x 4 (minor)
Items: Item1 to Item2
Major axis: 2000-01-09 00:00:00 to 2000-01-25 00:00:00
Minor axis: A to D
```

In [9]:
```
# here we select on the minor axis
slicepanel2 = store.select('appendpanel', where = [ dict(field = 'column',
slicepanel2
```

Out[9]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 30 (major) x 2 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-30 00:00:00
```

```
Minor axis: A to B
```

In [10]:
```
# delete operations
store.remove('appendpanel',where = [ dict(field = 'column', value = ['A','
store.select('appendpanel')
```

Out[10]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 30 (major) x 2 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2000-01-30 00:00:00
Minor axis: C to D
```

notes & caveats

seleting on both major & minor axis is possible (extend the where clause)
selection by items (top level panel dimension) is not possible; you always get all of the items in the returned panel
in general it is best to store your panel with the most frequently selected dimension in the minor axis and a time/date like dimension in the major axis
mixed type items are currently not supported (e.g. all of your data must be floats)
currently the major_axis is NOT indexed by pytables (as there is a bug in the pytables spec for this)

performance is quite good on the sub-selections and tables sizes can be quite large
in fact you can often append panels objects to create a giant table on disk, then subselect out as needed (e.g. write once - read many)

in general I compress tables after writing them (using blosc compression) - much slower if you compress as you go

If I am deleting a lot of data, I will either rebuild the table (erase and rewrite),
or use the pytables utilities ptrepack to rewrite the file (and also can change compression methods)

once a table is written, the items are fixed for that table; you can append only items that match exactly those on disk
(if you want to change this, then rebuild - e.g. erase and write a new table)

duplicate items can be written, but are filtered out in selection (with the last items being selected; thus a table is unique on major, minor pairs)

In [11]:
```
large_p = pandas.Panel(np.random.randn(2, 1000, 1000), items=['Item1', 'It
    major_axis=pandas.date_range('1/1/2000', periods=1000), minor_axis = [
large_p
```

Out[11]:
```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 1000 (major) x 1000 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2002-09-26 00:00:00
Minor axis: E0 to E999
```

In [12]:
```
if os.path.exists('large_store.h5'):
        os.remove('large_store.h5')
large_store = pandas.io.pytables.HDFStore('large_store.h5')
```

In [13]:

```
large_store.append('large',large_p)
```

In [17]:

```
# we basically wrote a structure of major_axis x minor_axis rows (with ite
print large_store.handle.root.large.table
```

```
/large/table (Table(1000000,)) ''
```

In [16]:

```
### on a slow machine!

def f():
    return large_store.select('large',where = [ dict(field = 'index', op =
print f(), "\n"

print "selection by major_axis"
%timeit f()

print "\n"
def f():
    return large_store.select('large',where = [ dict(field = 'column', val
print f(), "\n"

print "selection by minor axis"
%timeit f()
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 15 (major) x 1000 (minor)
Items: Item1 to Item2
Major axis: 2002-09-12 00:00:00 to 2002-09-26 00:00:00
Minor axis: E0 to E999

selection by major_axis
1 loops, best of 3: 607 ms per loop


<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 1000 (major) x 100 (minor)
Items: Item1 to Item2
Major axis: 2000-01-01 00:00:00 to 2002-09-26 00:00:00
Minor axis: E0 to E99

selection by minor axis
1 loops, best of 3: 1.1 s per loop
```

In [15]: