

Midas Data Miner, going forward (to-do items before Midas Backtester):

I need to refine my raw “primary” data into “secondary” data which is better suited for Midas Backtester. Here are some goals:

- 1) Eliminate excessive information (any quote attributes I won't use / need at the moment)
- 2) Cull bad tickers (any tickers for which I have insufficient market data)
- 3) Refine BTree into more efficient data structures for Midas Backtester performance
- 4) Flag existing secondary keys when they are updated in the primary data ... to speed up subsequent primary->secondary data imports
- 5) Devise a uniform quote attribute information scheme which still accommodates me to use a variety of primary data sources

Todo items:

“Hacks” -> “Prefix Sources to Attribute Keys” One-time script to rename all existing quote attributes to “source_PERatio” scheme.

- Modify existing import code to also include this suffix

“Hacks” -> “Add Yahoo Date Ranges” One-time script to add existing market data ranges to marketDataRange data structure

- Modify existing yahoo import code to also create these market data range values

“Secondary Data” -> “New / Delete Secondary Data” : script to add all primary keys to secondaryDirtyKeys set ::::: “Overwrite Secondary data” : deletes BTree, adds primary keys to secondaryDirtyKeys (include HARSH warning about danger).. deletes quoteAttributes data structures, etc.

- TO BE SAFE: Export quote data as a precaution.... (using that particular function)

“Secondary Data” -> “Import Primary Data” :

- 1) Imports all keys in secondaryDirtyKeys from primary data -> secondary data
- 2) Interactive script adds names to quoteAttributesAliases or quoteAttributesIgnored per user input (and quoteAttributesFancy)
- 3) Interactive script should have: “Go unattended?” every time a quote is successful imported without asking user for any information (e.g. every attribute already entered as alias or ignore-listed) --> in which case all subsequent data imports unattended per [quoteAttributes]

“Secondary Data” -> “Generate company series” : Generates companySeries data structs, including TRAILING FUNDAMENTAL DATA (mrq/mra)

“Secondary Data” -> “Generate day series” : Generates daySeries data structs, including TRAILING FUNDAMENTAL DATA (mrq/mra/mrd)

“Secondary Data” -> “Generate month series” : Generates daySeries data structs, including TRAILING FUNDAMENTAL DATA (mrq/mra/mrd)

- “mrd_” is most recent damadoran, but will probably develop into “most recent in last year” versus “most recent annual report” once I find future sources (or just become antiquated and removed from secondary data attributes)

“Secondary Data” -> “Erase ignore list” : Erase quoteAttributesIgnored list ... next import will query user for data again

“Secondary Data” -> “Export quote attributes” : Exports fancy quote attributes / short code names to a .txt file

ZODB Data Structures

Name	Data Structure	Key	Value	Description
primary	BTree	(date, ticker) tuple	Quote object	Raw data from sources
<i>primaryKeysAll</i> [deemed unnecessary]	<i>TreeSet</i>	<i>(date, ticker) tuple</i>	--	<i>Does NOT Persist:</i> <i>Created at start of REFINE process</i>
secondary	BTree	(date, ticker) tuple	Quote object	Refined data from primary BTree Special function performs: 1) culls bad companies 2) refines variety of raw data keys to uniform keys 3) only updates keys that need updating (see below)
<i>secondaryKeys</i> [deemed unnecessary]	<i>TreeSet</i>	<i>(date, ticker) tuple</i>	--	<i>Use: Speeds up REFINING function by ignoring keys that already exist.</i> <i>Used in conjunction with secondaryKeysDirty.</i> <i>See footnote [2] for use.</i> <i>Creation: Quotes added to secondary from primary are added to this set.</i> <i>(or otherwise created in total from secondary.keys() at any time)</i>
secondaryDirtyKeys	TreeSet	(date, ticker) tuple	--	See footnote [2] for use. Creation: Whenever primary data is modified, add that tuple to this set.
companySeries	BTree	(ticker)	PM Dict object * quotes keyed by date * "dates" set	Historical time series for company (w/ fundy data) Creation: Manual function, balances tree See footnote [3] for dirty implementation.
daySeries	BTree	(date)	PM Dict object * quotes keyed by ticker * "tickers" set	One day of data for all companies, all dates Creation: Manual function, balances tree See footnote [3] for dirty implementation
monthSeries	BTree (size small? use dict)	(month) or (date)	PM Dict object * quotes keyed by ticker * "tickers" set	One day of data for all companies, by months Creation: Manual function, balances tree See footnote [3] for dirty implementation See footnote [4] for rationale re: access time
quoteAttributes	Dict (PersistentMapping)	string	string	Fancy names for brief keys ("trpe" = "Trailing PE Ratio") See footnote [5]
quoteAttributeAliases	Dict (PersistentMapping)	string	string	Map raw field names ("damadoran_ PEratio") to brief keys ("mrd_ PEratio") etc.
quoteAttributesIgnored	Dict (PersistentMapping)	string	string	This structure keeps track of culled primary attributes (that are ignored when building secondary data)
quoteAttributesOrdered	List	string	--	Comprises quoteAttributesAliases keys - footnote [6]
marketDataRange	BTree	(ticker) string	(begin, end) tuple	See footnote [1]

1. It will save a lot of time to be able to ignore marketData that has already been collected. Small BTree contains tickers keyed to (begin,end) tuples which tell the range of already-imported market data (so next time I import yahoo data, I can ignore date ranges that have already been added)
2. Use:
 - a. (when modifying a primary key)
 - b. **secondaryDirtyKeys.add(tuple)**
 - c.
 - d. (when refining primary -> secondary BTree)
 - e. **for key in list(secondaryDirtyKeys):** [alternative: use secondaryKeysDirty.difference(secondaryKeys)]
 - f. **add/overwrite to secondary BTree # randomize for balanced tree?**
 - g. secondaryNew: This is flagged True in the session when the secondary BTree is created. Becomes flagged false at conclusion of first refine.
3. Dirty detection is NOT implemented:
 - a. Memory intensive to modify, since large dict objects must be loaded to memory entirely for any single key-value change.
 - b. Ergo, would throttle performance A LOT to force program to update these dicts on-the-fly when secondary data is refined.
 - c. Companies and Dates BTrees are created MANUALLY by the user before using backtester / other softwares. Very time intensive, but more helpful to create them only the one time before deployment to Midas Backtester rather than slowing down data mining.
 - d. Possible dirty detection in the future: Create companiesDirty and datesDirty tree sets to update only those modified keys in this batch call before deployment.
4. Months vs Days BTrees
 - a. I do not know, but it may be faster to write a historical time series of data by months-granularity for faster access if backtesting does not need daily/weekly granularity.
 - b. days BTree: 10 years x ~250 trading days x ~7000 companies = ~17.5 million keys
 - c. months BTree: 10 years x 12 months x ~7000 companies = 840 thousand keys
 - d. Practically, months is 1/20th smaller (as predicted by 12/250 ...). How much will this affect access time? I don't know. If binary tree is balanced, it's probably only a few more rungs to climb (not much time)... though if binary tree is UNbalanced, then this change helps a lot.
 - e. Also: pickling entire dicts to/from the hard disk may be better performance than navigating a BTree which likes to keep what it can stored to disk... it's better for persistence than performance, yeah?
5. I need to create a uniform naming convention for quote information
 - a. While this could be included in the script documentation (and not in the ZODB), I feel like it belongs in the ZODB in case the script is ever lost... otherwise you have a ZODB full of cryptic data referencing "trpe" and crap but not knowing what this information is.
 - b. Also: Instead of depending on sharing code between Midas Data Miner and Midas Backtester, I can rely on this dictionary to figure out things.
6. quoteAttributesOrdered will not be implemented until later... Clone of quoteAttributesAliases:
 - a. I re-order this dictionary to prefer certain sources over another... if "aaii_PERatio" comes before "damadoran_PERatio" and both write to "PERatio" in the final quote object, then damadoran prevails... Implicitly, this would mean that I believe that damadoran data is more credible.