Questions regarding implementation to address

1. What format to store the optionchain data in to make retrieval simplistic, bandwidth efficient, and automatable? Some alternatives:
   * 1. one sqlite3 database for each symbol, which has an underlying table and a linked options table that can be parsed by any software that understands sqlite3 tables. An example table definition is contained in the next section. This has the advantage of simplifying storage (one directory of symbols, each containing the entire historical record in a way that makes querries simple. Software experience required.
     2. a python pickle'd binary file for each symbol, for each trading day. A lot of us use python, so getting the data out is trivial. Restrictive to users that don't use python. Also adds the complexity of a directory structure that is now two levels deep. This is perhaps not a big deal. The top level directory could be the symbol, followed by dates containing each trading days option chain.
     3. much as above, but using a JSON file. This is the format of the original data from Yahoo or Google. Perhaps to save bandwidth and storage, a directory of JSON files for each symbol is stored as a 7zip archive. A JSON format allows different contributors to use different sources of data; Yahoo, Google, NSE all provide somewhat different data.
2. cloud storage. Many providers give away 5-20 GB of online storage. There are interfaces to all of these that would allow a user to rsync (bandwidth efficiently) updates to a directory full of symbol option chains. A useful tool to use may be https://rclone.org/, which looks to be cloud agnostic. If Alice used Dropbox to store her 100 symbols, but Fred used Amazon's S3 to store his 125 symbols, then anyone could access these public shares in anyway they wished, but we would encourage a bandwidth efficient method such as rclone, in order to prevent someone from getting kicked off their free cloud storage for excessive bandwidth usage. Any consumer of the historical data could thereby pick and choose the symbols and contributors they wanted to “subscribe” to; for example, three users could track different symbols, per the table below. It is trivial to see which symbols are being tracked by simply getting a list of the top level directory.

|  |  |  |
| --- | --- | --- |
| **Alice @ Dropbox/share** | **Fred @ AWS S3/share** | **Tom @ Gdrive/share** |
| AAPL.7z | GOOGL.7z | CRM.7z |
| SPY.7z | ABT.7z |  |
| NDX.7z | AMZN.7z |  |
| YUM.7z |  |  |
| ANF.7z |  |  |
| SFM.7z |  |  |

1. A decentralized approach: most investors concern themselves with a few hundred different symbols at any given time. Downloading options chains data for a few hundred symbols on a nominal home broadband connection takes several minutes. Keeping and sharing these datasets with others provides a larger audience and a larger historical record to be shared. We should encourage other contributors to first scan other users directories for symbols before creating another archive of the same symbol. How many users have to download the options chain for GOOGL each day? The nominative answer, of course, is one.

Current Sources of options chain data:

1. Yahoo Finance (<https://query1.finance.yahoo.com/v7/finance/options/GOOGL>)
2. NASDAQ (<https://www.nasdaq.com/symbol/GOOGL/option-chain>)
3. Google Finance (may be broken)

An example sqlite3 database for underlying and options chain. This example only contains a portion of the options chain information collected is ingested in these tables. This argues for using an alternative format that includes all the options chain data, such as a JSON file.

CREATE TABLE Underlying(

rowid INTEGER PRIMARY KEY AUTOINCREMENT,

utc\_timestamp INTEGER NOT NULL,

volume INTEGER NOT NULL,

AverageDailyVolume INTEGER NOT NULL,

bid REAL,

ask REAL,

last REAL NOT NULL,

Open REAL NOT NULL,

PreviousClose REAL NOT NULL,

PriceBook REAL,

PERatio REAL,

FiftydayMovingAverage REAL,

TwoHundreddayMovingAverage REAL,

YearHigh REAL,

YearLow REAL,

utc\_datestr TEXT DEFAULT NULL,

earningsTimestamp INTEGER DEFAULT NULL,

averageDailyVolume10Day INTEGER DEFAULT NULL,

averageDailyVolume3Month INTEGER DEFAULT NULL,

epsForward REAL DEFAULT NULL,

epsTrailingTwelveMonths REAL DEFAULT NULL,

forwardPE REAL DEFAULT NULL,

trailingPE REAL DEFAULT NULL);

CREATE TABLE Option(

rowid INTEGER PRIMARY KEY AUTOINCREMENT,

underlying\_id INTEGER NOT NULL,

expiration\_jday INTEGER NOT NULL,

strike\_price REAL NOT NULL,

type TEXT NOT NULL,

volume INTEGER NOT NULL,

open\_interest INTEGER NOT NULL,

last REAL NOT NULL,

bid REAL,

ask REAL,

expiration\_datestr TEXT DEFAULT NULL,

impliedVolatility REAL NOT NULL DEFAULT 0.0,

FOREIGN KEY(underlying\_id) REFERENCES Underlying(rowid) );

Example python code to download Yahoo options chain data:

import requests

import demjson

url = 'https://query1.finance.yahoo.com/v7/finance/options/GOOGL'

r = requests.get(url, timeout=20)

opt = demjson.decode(r.text)

underlying = opt['optionChain']['result'][0]['quote']

chain = opt['optionChain']['result'][0]['options'][0]

expiration\_seconds = opt['optionChain']['result'][0]['expirationDates']

then iterate over expiration\_seconds for the optionschains for other expiry months.