Crypto exchanges quotes harvester

Version: 1.11

Console utility to get quotes from 2 crypto-exchanges, generate quotes for synthetic instruments, and write all data into a database.

Requirements:

- Follow code conventions: https://bitbucket.org/FinPlant/public-documentation/wiki/Coding%20conventions or https://bit.ly/2oDc9Oj
- The application has to be fully workable and compiles without warnings
- All errors should be treated as critical, they have to be logged, and application should be closed
- Deadline is 2 weeks + extra time for review

Prerequisites:

After a candidate takes this test, we create for him:

- Private git repository in Bitbucket
- Private Slack channel

Workflow:

1. All code should be written in a feature branch
2. After work is done, pull request should be created and reviewed by us
3. After every comment will be fixed, the branch will be merged and test considered as done

Vocabulary

- Quote – a combination of bid and ask, for instance [100.25, 100.35]
- Bid – price to sell
- Ask – price to buy. Always bigger than a bid
- Instrument – a pair of currencies (base/counter), for instance, BTC/USD – bitcoin as a base currency per dollar as a counter currency.
- Synthetic instrument – instrument calculated based on other instruments. See the algorithm further.

Components

1. Exchanges endpoints (readers)

Each endpoint connects to one of the exchanges: **Binance** and **Polonex**

Each of them has the following functions:

1. Connects to specific exchange via “xchange-stream” library: https://github.com/FinPlant/xchange-stream.
2. To add it into your project use instruction from here: https://jitpack.io/#FinPlant/xchange-stream/develop-SNAPSHOT
3. Use modules “xchange-binance” and “xchange-poloniex2”
4. Subscribe to quotes stream for every instrument (except synthetics) defined in settings by using function `getTicker`
5. Store arrived quotes into a buffer

2. The calculator of synthetic instruments quotes (generator)

When the specific instrument in settings has sub-instruments (in settings section: “depends”), it’s assigned as **synthetic**. Its quotes not received from the exchange, but should be calculated using a combination of other “real” instruments and written into DB between other quotes.

See the calculation formula below.

3. Quotes buffer (in-memory storage)

Stores all recently arrived quotes. If an incoming quote has the same instrument and same exchange as already stored in the buffer, it should overwrite an old value.

4. Database endpoint (writer)

Connects to MySQL database, and write collected data from the buffer in a batch.

Application logic

Database and table defined in settings should be created at application startup. As an option, you can use some automatic tools like Docker with database creation script or Liquibase. See DDL below.

Then the application connects to both exchanges, subscribes to quotes for all regular instruments, and as soon as new data arrives, collects in the “write buffer”. Synthetic instruments also are calculated, based on received data and stored in the buffer also.

Each **flush_period_s** seconds (see settings), the writer saves all data into DB by a batch, and clears “write buffer”.

It could be a situation when at synthetics calculation moment, the application has no both quotes from instruments between arrived data. Then the missing quote has to be received from the cache. So actually there have to be 2 buffers with the same structure, one to read quotes for synthetics calculation (“read buffer”, aka “cache”), and the second one to have quotes for writing into DB, that periodically clears (“write buffer”).

Synthetic instrument calculation

Synthetic instrument is calculated that way:
base1/counter1 / base2/counter2 = *base1/base2*, where:

base1, counter1 – input instrument #1, for instance EUR/USDT
base2, counter2 – input instrument #2, for instance BTC/USDT
and counter1 == counter2

This calculation has to be done twice, for bid and ask

Example (for the bids only)

Given:

- BTC/EUR– bid: 7698
- USDT/EUR – bid: 0.84
- Output (aka synthetic): BTC/USDT

Then: BTC/USDT = BTC/EUR / USDT/EUR = 7698/0.84 = 9114

Table DDL

```
CREATE TABLE `QUOTES` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `TIME` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `BID` decimal(20,8) NOT NULL,
  `ASK` decimal(20,8) NOT NULL,
  `EXCHANGE` varchar(20) NOT NULL,
  `NAME` varchar(20) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `QUOTES_TIME_IDX` (`TIME`) USING BTREE,
  KEY `QUOTES_SYNTHETIC_IDX` (`NAME`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8
```

Settings example (YAML)

```
---
db:
  url: …
  user: ...
  password: …
flush_period_s: …
instruments:
  - name: BTCUSDT
    instrument: BTC/USDT
  - name: ETHUSDT
    instrument: ETH/USDT
  - name: BTCUSD-synth1
    instrument: BTC/USDT
    depends:
      - ETH/BTC
      - ETH/USDT
  - name: BTCUSDT-synth2
    instrument: BTC/USDT
    depends:
      - XRP/BTC
      - XRP/USDT
```

Each instrument has a name. It has to be unique and has to be written into the table "NAME" field.