- 1. 分钟线时间区间
- 2. 计算逻辑](#2-计算逻辑)
  - 2.1. 源数据特点](#21-源数据特点)
  - 2.2. 特殊情况及其处理](#22-特殊情况及其处理)
  - 2.3. 字段计算逻辑详解](#23-字段计算逻辑详解)
    - 2.3.1. 基于tick的字段](#231-基于tick的字段)
    - 2.3.2. 基于逐笔的字段](#232-基于逐笔的字段)
    - 2.3.3. 统计量定义](#233-统计量定义)
      - 2.3.3.1. $mid\_price$(#2331-mid_price)
      - 2.3.3.2. $spread$(#2332-spread)
      - 2.3.3.3 $delta\_amount\_ask$(#2333-delta_amount_ask)
      - 2.3.3.4 $delta\_amount\_bid$(#2334-delta_amount_bid)
      - 2.3.3.5 $qimb$(#2335-qimb)
      - 2.3.3.6 $tick\_return$(#2336-tick_return)
      - 2.3.3.7 $book\_ratio$(#2337-book_ratio)
      - 2.3.3.8 $book\_rratio$(#2338-book_rratio)
      - 2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount)
- 3. 后续版本需要解决的问题](#3-后续版本需要解决的问题)
  - 3.1.加入更多基于逐笔的字段](#31加入更多基于逐笔的字段)
  - 3.2. 解决15点47分集合竞价时tick的判定问题](#32-解决15点47分集合竞价时tick的判定问题)
  - 3.3. 对没收到tick的分钟设置一个特殊流程](#33-对没收到tick的分钟设置一个特殊流程)
  - 3.4. 盘后真实还原盘中时钟](#34-盘后真实还原盘中时钟)
  - 3.5. 每只票收齐则立刻发送分钟线](#35-每只票收齐则立刻发送分钟线)
  - 3.6. 考虑加入盘中拉起的功能](#36-考虑加入盘中拉起的功能)
  - 3.7. 优化掉性能瓶颈](#37-优化掉性能瓶颈)


# 1. 分钟线时间区间

为了兼容之前的版本，规定`bar_start_time`等于`bar_end_time`减去一分钟。

下表中`sp`是一个阈值，目前在`v0.10.0`中为`30`。

| bar_start_time | bar_end_time | 实际对应交易所时间区间 | 需特殊关注 |
|---|---|---|---|
| 09:24:00 | 09:25:00 | [09:15:00,09:30:00) | √ |
| 09:30:00 | 09:31:00 | (09:30:00,09:31:00] | √ |
| 09:31:00 | 09:32:00 | (09:31:00,09:32:00] | |
| 09:32:00 | 09:33:00 | (09:32:00,09:33:00] | |
| ... | ... | ... | |
| 11:28:00 | 11:29:00 | (11:28:00,11:29:00] | |
| 11:29:00 | 11:30:00 | (11:29:00,11:30:sp] | √ |
| 13:00:00 | 13:01:00 | [13:00:00,13:01:00] | √ |
| 13:01:00 | 13:02:00 | (13:01:00,13:02:00] | |
| 13:02:00 | 13:03:00 | (13:02:00,13:03:00] | |
| 13:03:00 | 13:04:00 | (13:03:00,13:04:00] | |
| ... | ... | ... | |
| 14:57:00 | 14:58:00 | (14:57:00,14:58:00] | |
| 14:58:00 | 14:59:00 | (14:58:00,14:59:00] | |
| 14:59:00 | 15:00:00 | (14:59:00,15:00:sp] | √ |

# 2. 计算逻辑

## 2.1. 源数据特点

分钟线是从tick数据/逐笔成交数据算出来的，为了解释分钟线数据中某些现象，需要介绍源数据(tick数据和逐笔成交数据)的特点。

1. 理想情况下每隔3s就会收到一个tick，但实际情况中为了节约带宽，如果某个时间点上tick的所有信息和上一个tick一模一样，交易所会省略这个tick。因此每分钟实际收到的tick个数可能小于20个，两个tick之间的间隔也可能大于3s。对于交易不活跃的股票，可能某个分钟内不会收到任何tick。
2. 有时候由于行情变化太快，交易所有可能增加发送tick的频率，造成两个tick之间的间隔小于3s，也就是说此分钟的tick个数大于20个。这种情况在接近收盘时容易出现。
3. 逐笔成交数据在某些时候的时延很大(主要是集合竞价之后)，可能达到20s以上。tick的时延一般不会超过3s。
4. tick数据和逐笔数据是两个异步的流，网络波动可能会导致其中一种数据暂时中断，而另一个种数据仍然正常接受。此时基于逐笔的分钟线字段和基于tick的分钟线字段可能会算出不相符的数值(比如`total_trades_from_tick`=0而`total_trades_from_trans`>0)。这种情况较为罕见，但历史数据显示其可能存在。

## 2.2. 特殊情况及其处理

1. 如果某分钟没有收到过tick，则拿之前最新收到的一个tick补20次，然后再开始相关计算(解决2.1. 源数据特点](#21-源数据特点)第1条)。因此在2.3. 字段计算逻辑详解](#23-字段计算逻辑详解)中将假设(第一次收到tick之后)每分钟至少存在一个`tick`。
2. 基于逐笔成交的字段和基于tick的字段分开计算，分开发送。
3. 自每天第一次收到某只股票的行情之后，才开始计算并推送这只股票的分钟线。自收到某股票第一个行情信息的那一刻起，产生的分钟bar都是时间上连续的(即不会中间缺失一个bar)。
4. 上述2.1. 源数据特点](#21-源数据特点)第4条可能导致基于tick的字段和基于逐笔的字段算出来的分钟bar条数不同。此时，如果要把基于tick的字段和基于逐笔的字段合并成一张表入库，就会存在"某个基于tick的分钟bar找不到对应时间的逐笔bar"的情况。如果遇到这种情况，则把该行的`arrival_time_from_trans`填为`1970-01-01 00:00:00`，这行所有基于逐笔的字段的浮点数填`NAN`，整数填`0`，字符串填空串。反之亦然。

## 2.3. 字段计算逻辑详解

### 2.3.1. 基于tick的字段

阅读前请注意：

- 下表中"计算方法"此列只描述计算逻辑，并不代表实际程序。但可以保证实际程序算出来的结果和"计算方法"所描述的结果相同。

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
|---|---|---|---|---|---|
| 1 | bopu_symbol | string | 形如`000001\|ST\|SZSE` | | |
| 2 | trade_date | datetime | 交易日 | | |
| 3 | bar_start_time | datetime | 分钟线的字面结束时间(与实际时间的对应关系见`分钟线时间区间`部分) | | |
| 4 | bar_end_time | datetime | 分钟线的字面结束时间(与实际时间的对应关系见`分钟线时间区间`部分) | | |
| 5 | data_source | string | 用来计算分钟线的行情数据源 | | |
| 6 | arrival_time_from_tick | datetime | 可近似认为是发出时间 | 这根bar开始进行最后一步计算(final_prorcess)时，当前已收到行情中bdf_timestamp的最大值 | |
| 7 | open_from_tick | float64 | bar内开盘价 | 第一个tick的`last_price` | 如果今天还没有发生交易，则此项的值为昨收价 |
| 8 | close_from_tick | float64 | bar内收盘价 | 最后一个tick的`last_price` | 如果今天还没有发生交易，则此项的值为昨收价 |
| 9 | high_from_tick | float64 | bar内最高价 | 此分钟内所有`last_price`的最大值 | 如果今天还没有发生交易，则此项的值为昨收价 |
| 10 | low_from_tick | float64 | bar内最低价 | 此分钟内所有`last_price`的最小值 | 如果今天还没有发生交易，则此项的值为昨收价 |
| 11 | high_to_now_from_tick | float64 | bar右端点的当日最高价 | 此分钟内以及之前收到的tick中所有"今日最高价"的最大值 | |
| 12 | low_to_now_from_tick | float64 | bar右端点的当日最低价 | 此分钟内以及之前收到的tick中所有"今日最低价"的最小值 | |
| 13 | accvolume_from_tick | int64 | bar右端点的当日累计成交量 | 此分钟内收到的最后一个tick中的累计成交量 | |
| 14 | volume_from_tick | int64 | bar内成交量 | 这根bar的accvolume_from_tick减去前一根bar的accvolume_from_tick | |
| 15 | accamount_from_tick | float64 | bar右端点的当日累计成交额 | 此分钟内收到的最后一个tick中的累计成交额 | |
| 16 | amount_from_tick | float64 | bar内成交额 | 这根bar的acamount_from_tick减去上一根bar的accamount_from_tick | |
| 17 | open_amount_from_tick | float64 | bar内第一个tick到其前一个tick之间的成交额 | bar内第一个tick的累计成交额减去上一根bar最后一个tick的累计成交额 | |
| 18 | close_amount_from_tick | float64 | bar内最后一个tick到其前一个tick之间的成家额 | bar内最后一个tick的累计成交额减去倒数第二个tick的累计成交额(如果无倒数第二个，就用上一根bar最后一个) | |
| 19 | high_amount_from_tick | float64 | bar内所有tick成交额的最大值 | bar每个tick的累计成交额和其前一个tick的累计成交额做差，得到的多个差值中求最大值 | |
| 20 | low_amount_from_tick | float64 | bar内所有tick成交额的最小值 | bar每个tick的累计成交额和其前一个tick的累计成交额做差，得到的差值中求最小值 | |
| 21 | iopv_from_tick | float64 | 未定义 | 目前全填0 | |
| 22 | acc_total_trades_from_tick | int64 | bar右端点的当日累计成交笔数 | 最后一个tick的当日累计成交笔数 | |
| 23 | total_trades_from_tick | int64 | bar内累计成交笔数 | 此bar的acc_total_trades_from_tick减去上一根bar的acc_total_trades_from_tick | |
| 24 | open_ask1_price_from_tick | float64 | bar内首个卖一价 | bar内第一个存在ask1的tick的卖一价 | 如果bar内所有tick都没有ask1，则用前一个bar的close_ask1_price |
| 25 | open_ask1_size_from_tick | int64 | bar内首个卖一量 | bar内第一个存在ask1的tick的卖一量 | 如果bar内所有tick都没有ask1，则此字段填0 |
| 26 | open_bid1_price | float64 | bar内首个买一价 | bar内第一个存在bid1的tick的买一价 | 如果bar内所有的tick都没有bid1，就用上一个bar的close_bid1_price |
| 27 | open_bid1_size | int64 | bar内首个买一量 | bar内第一个存在bid1的tick的买一量 | 如果bar内所有的tick都没有bid1，则填0 |
| 28 | close_ask1_price | float64 | bar最后一个卖一价 | bar内最后一个存在ask1的tick的卖一价 | 如果bar内所有tick都没有ask1，则用前一个bar的close_ask1_price |
| 29 | close_ask1_size | int64 | bar内最后一个卖一量 | bar内最后一个存在ask1的tick的卖一量 | 如果bar内所有tick都没有ask1，则填0 |
| 30 | close_bid1_price | float64 | bar最后一个买一价 | bar内最后一个存在bid1的tick的买一价 | 如果bar内所有tick都没有bid1，则用前一个bar的close_bid1_price |
| 31 | close_bid1_size | int64 | bar内最后一个买一量 | bar内最后一个存在bid1的tick的买一量 | 如果bar内所有tick都没有bid1，则填0 |
| 32 | high_ask1_price_from_tick | float64 | bar内最高卖一价 | bar内所有存在ask1的tick的ask1价格的最大值 | 如果bar内所有tick都没有ask1，取前一个bar的close_ask1_price |
| 33 | high_ask1_size_from_tick | int64 | bar内最高卖一价所对应tick的卖一量 | 找到所有ask1价格等于high_ask1_price_from_tick的tick，对这些tick的ask1量取平均，再四舍五入到整数 | 如果bar内所有tick都没有ask1，填0 |
| 34 | high_bid1_price_from_tick | float64 | bar内最高买一价 | bar内所有存在bid1的tick的bid1价格的最大值 | 如果bar内所有tick都没有bid1，取前一个bar的close_bid1_price |
| 35 | high_bid1_size_from_tick | int64 | bar内最高买一价所对应tick的买一量 | 找到所有bid1价格等于high_bid1_price_from_tick的tick，对这些tick的bid1量取平均，再四舍五入到整数 | 如果bar内所有tick都没有bid1，填0 |

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
|---|---|---|---|---|---|
| 36 | low_ask1_price_from_tick | float64 | bar内最低卖一价 | bar内所有存在ask1的tick的ask1价格的最小值 | 如果bar内所有tick都没有ask1，取前一个bar的close_ask1_price |
| 37 | low_ask1_size_from_tick | int64 | bar内最低卖一价所对应tick的卖一量 | 找到所有ask1价格等于low_ask1_price_from_tick的tick，对这些tick的ask1量取平均，再四舍五入到整数 | 如果bar内所有tick都没有ask1，填0 |
| 38 | low_bid1_price_from_tick | float64 | bar内最低买一价 | bar内所有存在bid1的tick的bid1价格的最小值 | 如果bar内所有tick都没有bid1，取前一个bar的close_bid1_price |
| 39 | low_bid1_size_from_tick | int64 | bar内最低买一价所对应tick的买一量 | 找到所有bid1价格等于low_bid1_price_from_tick的tick，对这些tick的bid1量取平均，再四舍五入到整数 | 如果bar内所有tick都没有bid1，填0 |
| 40 | avg_ask1_price_from_tick | float64 | bar内平均卖一价 | bar内所有卖一价的平均值(不考虑无卖1档的tick) | 如果bar内所有tick都没有ask1，取前一个bar的close_ask1_price |
| 41 | avg_ask1_size_from_tick | int64 | bar内平均卖一量 | bar内所有卖一量的平均值(不考虑无卖1档的tick)，保留整数 | 如果bar内所有tick都没有ask1，填0 |
| 42 | avg_bid1_price_from_tick | float64 | bar内平均买一价 | bar内所有买一价的平均值(不考虑无买1档的tick) | 如果bar内所有tick都没有bid1，取前一个bar的close_bid1_price |
| 43 | avg_bid1_size_from_tick | int64 | bar内平均买一量 | bar内所有买一量的平均值(不考虑无买1档的tick)，保留整数 | 如果bar内所有tick都没有bid1，填0 |
| 44 | vwap_ask1_price_from_tick | float64 | bar内所有tick的卖一价，以卖一量为权值的加权平均 | bar内所有tick的卖一价，以卖一量为权值的加权平均(不考虑无卖一档的tick) | 如果bar内所有tick都没有ask1，取前一个bar的close_ask1_price |
| 45 | vwap_bid1_price_from_tick | float64 | bar内所有tick的买一价，以买一量为权值的加权平均 | bar内所有tick的买一价，以买一量为权值的加权平均(不考虑无买一档的tick) | 如果bar内所有tick都没有bid1，取前一个bar的close_bid1_price |
| 46 | open_mid_price_from_tick | float64 | bar内首个mid_price | 对bar内每一个tick都尝试计算mid，算出来的多个mid中取第一个值(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内所有tick都算不出mid，则此字段填nan |
| 47 | close_mid_price_from_tick | float64 | bar内最后一个mid_price | 对bar内每一个tick都尝试计算mid，算出来的多个mid中取最后一个值(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内所有tick都算不出mid，则此字段填nan |
| 48 | mid_price_avg_from_tick | float64 | bar内mid_price的均值 | 对bar内每一个tick都尝试计算mid，算出来的多个mid取均值(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内所有tick都算不出mid，则此字段填nan |
| 49 | mid_price_std_from_tick | float64 | bar内所有mid_price的标准差 | 对bar内每一个tick都尝试计算mid，算出来的多个mid求std(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内能算出mid的tick的个数小于2，则此字段填nan |
| 50 | mid_price_skew_from_tick | float64 | bar内所有mid_price的偏度 | 对bar内每一个tick都尝试计算mid，算出来的多个mid求skew(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内能算出mid的tick的个数小于3，则此字段填nan |
| 51 | mid_price_kurt_from_tick | float64 | bar内所有mid_price的峰度 | 对bar内每一个tick都尝试计算mid，算出来的多个mid求kurt(具体规则参照2.3.3.1. $mid\_price$(#2331-mid_price)) | 如果bar内能算出mid的tick的个数小于4，则此字段填nan |
| 52 | min_spread_from_tick | float64 | bar内所有spread的最小值 | 对bar内每一个tick都尝试计算spread，算出来的多个spread取最小值(具体规则参照2.3.3.2. $spread$(#2332-spread) | 如果bar内所有tick都算不出spread，则此字段填nan |
| 53 | max_spread_from_tick | float64 | bar内所有spread的最大值 | 对bar内每一个tick都尝试计算spread，算出来的多个spread取最大值(具体规则参照2.3.3.2. $spread$(#2332-spread) | 如果bar内所有tick都算不出spread，则此字段填nan |
| 54 | avg_spread_from_tick | float64 | bar内所有spread的均值 | 对bar内每一个tick都尝试计算spread，算出来的多个spread取均值(具体规则参照2.3.3.2. $spread$(#2332-spread) | 如果bar内所有tick都算不出spread，则此字段填nan |
| 55 | open_ask_amount10_from_tick | float64 | 分钟内首个tick卖前十档的amount之和 | 对bar内每个tick，计算卖前十档amount之和(跳过无ask1的tick)，取第一个tick的结果作为此字段 | 若所有tick都被跳过，则填0 |
| 56 | open_bid_amount10_from_tick | float64 | 分钟内首个tick买前十档的amount之和 | 对bar内每个tick，计算买前十档amount之和(跳过无bid1的tick)，取第一个tick的结果作为此字段 | 若所有tick都被跳过，则填0 |
| 57 | close_ask_amount10_from_tick | float64 | 分钟内最后一个tick卖前十档的amount之和 | 对bar内每个tick，计算卖前十档amount之和(跳过无ask1的tick)，取最后一个tick的结果作为此字段 | 若所有tick都被跳过，则填0 |
| 58 | close_bid_amount10_from_tick | float64 | 分钟内最后一个tick买前十档的amount之和 | 对bar内每个tick，计算买前十档amount之和(跳过无bid1的tick)，取最后一个tick的结果作为此字段 | 若所有tick都被跳过，则填0 |
| 59 | avg_ask_amount10_from_tick | float64 | 分钟内每个tick的"卖前十档amount之和"求平均 | 对bar内每个tick，计算卖前十档amount之和(跳过无ask1的tick)，得到的所有结果求均值 | 若所有tick都被跳过，则填0 |
| 60 | avg_bid_amount10_from_tick | float64 | 分钟内每个tick的"买前十档amount之和"求平均 | 对bar内每个tick，计算买前十档amount之和(跳过无bid1的tick)，得到的所有结果求均值 | 若所有tick都被跳过，则填0 |
| 61 | ask_volume10_avg_from_tick | float64 | 分钟内每个tick的"卖前十档volume之和"求平均 | 对bar内每个tick，计算卖前十档volume之和(跳过无ask1的tick)，得到的所有结果求均值 | 若所有tick都被跳过，则填0 |
| 62 | bid_volume10_avg_from_tick | float64 | 分钟内每个tick的"买前十档volume之和"求平均 | 对bar内每个tick，计算买前十档volume之和(跳过无bid1的tick)，得到的所有结果求均值 | 若所有tick都被跳过，则填0 |
| 63 | open_vwap_ask_price10_from_tick | float64 | 分钟内首个tick以委托量为权值对前十档卖价求加权平均 | 分钟内首个tick以委托量为权值对前十档卖价求加权平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_ask_price10_from_tick |
| 64 | open_avg_ask_price10_from_tick | float64 | 分钟内首个tick前十档卖价的算术平均 | 分钟内首个tick前十档卖价的算术平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一根bar的close_avg_ask_price10_from_tick |
| 65 | open_vwap_bid_price10_from_tick | float64 | 分钟内首个tick以委托量为权值对前十档买价求加权平均 | 分钟内首个tick以委托量为权值对前十档买价求加权平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_bid_price10_from_tick |
| 66 | open_avg_bid_price10_from_tick | float64 | 分钟内首个tick前十档买价的算术平均 | 分钟内首个tick前十档买价的算术平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一根bar的close_avg_bid_price10_from_tick |
| 67 | close_vwap_ask_price10_from_tick | float64 | 分钟内最后一个tick以委托量为权值对前十档卖价求加权平均 | 分钟内最后一个tick以委托量为权值对前十档卖价求加权平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_ask_price10_from_tick |

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
|---|---|---|---|---|---|
| 68 | close_avg_ask_price10_from_tick | float64 | 分钟内最后一个tick前十档卖价的算术平均 | 分钟内最后一个tick前十档卖价的算术平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一根bar的close_avg_ask_price10_from_tick |
| 69 | close_vwap_bid_price10_from_tick | float64 | 分钟内最后一个tick以委托量为权值对前十档买价求加权平均 | 分钟内最后一个tick以委托量为权值对前十档买价求加权平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_bid_price10_from_tick |
| 70 | close_avg_bid_price10_from_tick | float64 | 分钟内最后一个tick前十档买价的算术平均 | 分钟内最后一个tick前十档买价的算术平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一根bar的close_avg_bid_price10_from_tick |
| 71 | vwap_ask_price10_avg_from_tick | float64 | 分钟内每个tick以委托量为权值对前十档卖价求加权平均，所有tick算出的结果再求算术平均 | 分钟内每个tick以委托量为权值对前十档卖价求加权平均，所有tick算出的结果再求算术平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_ask_price10_from_tick |
| 72 | avg_ask_price10_avg_from_tick | float64 | 分钟内每个tick对前十档卖价求算术平均，所有tick算出的结果再求算术平均 | 分钟内每个tick对前十档卖价求算术平均，所有tick算出的结果再求算术平均(跳过无ask1的tick) | 若所有tick都被跳过，则填前一个bar的close_avg_ask_price10_from_tick |
| 73 | vwap_bid_price10_avg_from_tick | float64 | 分钟内每个tick以委托量为权值对前十档买价求加权平均，所有tick算出的结果再求算术平均 | 分钟内每个tick以委托量为权值对前十档买价求加权平均，所有tick算出的结果再求算术平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一个bar的close_vwap_bid_price10_from_tick |
| 74 | avg_bid_price10_avg_from_tick | float64 | 分钟内每个tick对前十档买价求算术平均，所有tick算出的结果再求算术平均 | 分钟内每个tick对前十档买价求算术平均，所有tick算出的结果再求算术平均(跳过无bid1的tick) | 若所有tick都被跳过，则填前一个bar的close_avg_bid_price10_from_tick |
| 75 | delta_amount_ask_algo1_from_tick | float64 | 基于tick计算delta_amount_ask_algo1，仅考虑委托簿前10档 | 详见2.3.3.3 $delta\_amount\_ask$(#2333-delta_amount_ask) | |
| 76 | delta_amount_bid_algo1_from_tick | float64 | 基于tick计算delta_amount_bid_algo1，仅考虑委托簿前10档 | 详见2.3.3.4 $delta\_amount\_bid$(#2334-delta_amount_bid) | |
| 77 | delta_amount_ask_algo2_from_tick | float64 | 基于tick计算delta_amount_ask_algo2，仅考虑委托簿前10档 | 详见2.3.3.3 $delta\_amount\_ask$(#2333-delta_amount_ask) | |
| 78 | delta_amount_bid_algo2_from_tick | float64 | 基于tick计算delta_amount_bid_algo2，仅考虑委托簿前10档 | 详见2.3.3.4 $delta\_amount\_bid$(#2334-delta_amount_bid) | |
| 79 | delta_amount_ask_algo3_from_tick | float64 | 基于tick计算delta_amount_ask_algo3，仅考虑委托簿前10档 | 详见2.3.3.3 $delta\_amount\_ask$(#2333-delta_amount_ask) | |
| 80 | delta_amount_bid_algo3_from_tick | float64 | 基于tick计算delta_amount_bid_algo3，仅考虑委托簿前10档 | 详见2.3.3.4 $delta\_amount\_bid$(#2334-delta_amount_bid) | |
| 81 | delta_amount_ask_algo4_from_tick | float64 | 基于tick计算delta_amount_ask_algo4，仅考虑委托簿前10档 | 详见2.3.3.3 $delta\_amount\_ask$(#2333-delta_amount_ask) | |
| 82 | delta_amount_bid_algo4_from_tick | float64 | 基于tick计算delta_amount_bid_algo4，仅考虑委托簿前10档 | 详见2.3.3.4 $delta\_amount\_bid$(#2334-delta_amount_bid) | |
| 83 | qimb1_avg_from_tick | float64 | 对每个tick求qimb1，分钟内所有的qimb1求avg | qimb1的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 84 | qimb1_std_from_tick | float64 | 对每个tick求qimb1，分钟内所有的qimb1求std | qimb1的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 85 | qimb1_skew_from_tick | float64 | 对每个tick求qimb1，分钟内所有的qimb1求skew | qimb1的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 86 | qimb1_kurt_from_tick | float64 | 对每个tick求qimb1，分钟内所有的qimb1求kurt | qimb1的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 87 | qimb10_avg_from_tick | float64 | 对每个tick求qimb10，分钟内所有的qimb10求avg | qimb10的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 88 | qimb10_std_from_tick | float64 | 对每个tick求qimb10，分钟内所有的qimb10求std | qimb10的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 89 | qimb10_skew_from_tick | float64 | 对每个tick求qimb10，分钟内所有的qimb10求skew | qimb10的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 90 | qimb10_kurt_from_tick | float64 | 对每个tick求qimb10，分钟内所有的qimb10求kurt | qimb10的定义见2.3.3.5 $qimb$(#2335-qimb) | |
| 91 | tick_return_avg_from_tick | float64 | bar内所有tick_return的均值 | 对每个tick计算tick_return，跳过无法计算tick_return的tick后求avg，tick_return的定义见2.3.3.6 $tick\_return$(#2336-tick_return) | 如果此bar内所有tick都被跳过，此字段填nan |
| 92 | tick_return_std_from_tick | float64 | bar内所有tick_return的std | 对每个tick计算tick_return，跳过无法计算tick_return的tick后求std，tick_return的定义见2.3.3.6 $tick\_return$(#2336-tick_return) | 如果跳过后剩余tick个数<2，此字段填nan |
| 93 | tick_return_skew_from_tick | float64 | bar内所有tick_return的std | 对每个tick计算tick_return，跳过无法计算tick_return的tick后求skew，tick_return的定义见2.3.3.6 $tick\_return$(#2336-tick_return) | 如果跳过后剩余tick个数<3，此字段填nan |
| 94 | tick_return_kurt_from_tick | float64 | bar内所有tick_return的std | 对每个tick计算tick_return，跳过无法计算tick_return的tick后求kurt，tick_return的定义见2.3.3.6 $tick\_return$(#2336-tick_return) | 如果跳过后剩余tick个数<4，此字段填nan |

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
|---|---|---|---|---|---|
| 95 | ask_amount10_chg_avg_from_tick | float64 | bar内卖前十档amount之和的变化量的avg | 每个tick的卖前十档之和与前一个tick的卖前十档之和做差，得到的样本集再求avg | |
| 96 | ask_amount10_chg_std_from_tick | float64 | bar内卖前十档amount之和的变化量的std | 每个tick的卖前十档之和与前一个tick的卖前十档之和做差，得到的样本集再求std | |
| 97 | ask_amount10_chg_skew_from_tick | float64 | bar内卖前十档amount之和的变化量的skew | 每个tick的卖前十档之和与前一个tick的卖前十档之和做差，得到的样本集再求skew | |
| 98 | ask_amount10_chg_kurt_from_tick | float64 | bar内卖前十档amount之和的变化量的kurt | 每个tick的卖前十档之和与前一个tick的卖前十档之和做差，得到的样本集再求kurt | |
| 99 | bid_amount10_chg_avg_from_tick | float64 | bar内买前十档amount之和的变化量的avg | 每个tick的买前十档之和与前一个tick的买前十档之和做差，得到的样本集再求avg | |
| 100 | bid_amount10_chg_std_from_tick | float64 | bar内买前十档amount之和的变化量的std | 每个tick的买前十档之和与前一个tick的买前十档之和做差，得到的样本集再求std | |
| 101 | bid_amount10_chg_skew_from_tick | float64 | bar内买前十档amount之和的变化量的skew | 每个tick的买前十档之和与前一个tick的买前十档之和做差，得到的样本集再求skew | |
| 102 | bid_amount10_chg_kurt_from_tick | float64 | bar内买前十档amount之和的变化量的kurt | 每个tick的买前十档之和与前一个tick的买前十档之和做差，得到的样本集再求kurt | |
| 103 | ask_amount10_ratio1_avg_from_tick | float64 | bar内ask_amount10序列的差分，每个值用前一个ask_amount10来nomalize，再求avg | bar内ask_amount10序列的差分，每个值除以前一个位置的ask_amount10，丢弃因分母为0而算出的非有限值，再求avg | 求avg时若样本集为空集，此字段填nan |
| 104 | ask_amount10_ratio1_std_from_tick | float64 | bar内ask_amount10序列的差分，每个值用前一个ask_amount10来nomalize，再求avg | bar内ask_amount10序列的差分，每个值除以前一个位置的ask_amount10，丢弃因分母为0而算出的非有限值，再求std | 求std时若样本数<2，此字段填nan |
| 105 | ask_amount10_ratio1_skew_from_tick | float64 | bar内ask_amount10序列的差分，每个值用前一个ask_amount10来nomalize，再求avg | bar内ask_amount10序列的差分，每个值除以前一个位置的ask_amount10，丢弃因分母为0而算出的非有限值，再求skew | 求skew时若样本数<3，此字段填nan |
| 106 | ask_amount10_ratio1_kurt_from_tick | float64 | bar内ask_amount10序列的差分，每个值用前一个ask_amount10来nomalize，再求avg | bar内ask_amount10序列的差分，每个值除以前一个位置的ask_amount10，丢弃因分母为0而算出的非有限值，再求kurt | 求kurt时若样本数<4，此字段填nan |
| 107 | bid_amount10_ratio1_avg_from_tick | float64 | bar内bid_amount10序列的差分，每个值用前一个bid_amount10来nomalize，再求avg | bar内bid_amount10序列的差分，每个值除以前一个位置的bid_amount10，丢弃因分母为0而算出的非有限值，再求avg | 求avg时若样本集为空集，此字段填nan |
| 108 | bid_amount10_ratio1_std_from_tick | float64 | bar内bid_amount10序列的差分，每个值用前一个bid_amount10来nomalize，再求avg | bar内bid_amount10序列的差分，每个值除以前一个位置的bid_amount10，丢弃因分母为0而算出的非有限值，再求std | 求std时若样本数<2，此字段填nan |
| 109 | bid_amount10_ratio1_skew_from_tick | float64 | bar内bid_amount10序列的差分，每个值用前一个bid_amount10来nomalize，再求avg | bar内bid_amount10序列的差分，每个值除以前一个位置的bid_amount10，丢弃因分母为0而算出的非有限值，再求skew | 求skew时若样本数<3，此字段填nan |
| 110 | bid_amount10_ratio1_kurt_from_tick | float64 | bar内bid_amount10序列的差分，每个值用前一个bid_amount10来nomalize，再求avg | bar内bid_amount10序列的差分，每个值除以前一个位置的bid_amount10，丢弃因分母为0而算出的非有限值，再求kurt | 求kurt时若样本数<4，此字段填nan |
| 111 | ask_amount10_ratio1_avg_from_tick | float64 | bar内ask_amount10序列的差分，每个值用当前tick的ask_amount10来nomalize，再求avg | bar内ask_amount10序列的差分，每个值除以当前tick的ask_amount10，丢弃因分母为0而算出的非有限值，再求avg | 求avg时若样本集为空集，此字段填nan |
| 112 | ask_amount10_ratio1_std_from_tick | float64 | bar内ask_amount10序列的差分，每个值用当前tick的ask_amount10来nomalize，再求std | bar内ask_amount10序列的差分，每个值除以当前tick的ask_amount10，丢弃因分母为0而算出的非有限值，再求std | 求std时若样本数<2，此字段填nan |
| 113 | ask_amount10_ratio1_skew_from_tick | float64 | bar内ask_amount10序列的差分，每个值用当前tick的ask_amount10来nomalize，再求skew | bar内ask_amount10序列的差分，每个值除以当前tick的ask_amount10，丢弃因分母为0而算出的非有限值，再求skew | 求skew时若样本数<3，此字段填nan |
| 114 | ask_amount10_ratio1_kurt_from_tick | float64 | bar内ask_amount10序列的差分，每个值用当前tick的ask_amount10来nomalize，再求kurt | bar内ask_amount10序列的差分，每个值除以当前tick的ask_amount10，丢弃因分母为0而算出的非有限值，再求kurt | 求kurt时若样本数<4，此字段填nan |
| 115 | bid_amount10_ratio1_avg_from_tick | float64 | bar内bid_amount10序列的差分，每个值用当前tick的bid_amount10来nomalize，再求avg | bar内bid_amount10序列的差分，每个值除以当前tick的bid_amount10，丢弃因分母为0而算出的非有限值，再求avg | 求avg时若样本集为空集，此字段填nan |
| 116 | bid_amount10_ratio1_std_from_tick | float64 | bar内bid_amount10序列的差分，每个值用当前tick的bid_amount10来nomalize，再求std | bar内bid_amount10序列的差分，每个值除以当前tick的bid_amount10，丢弃因分母为0而算出的非有限值，再求std | 求std时若样本数<2，此字段填nan |

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
|---|---|---|---|---|---|
| 117 | bid_amount10_ratio1_skew_from_tick | float64 | bar内bid_amount10序列的差分，每个值用当前tick的bid_amount10来nomalize，再求skew | bar内bid_amount10序列的差分，每个值除以当前tick的bid_amount10，丢弃因分母为0而算出的非有限值，再求skew | 求skew时若样本数<3，此字段填nan |
| 118 | bid_amount10_ratio1_kurt_from_tick | float64 | bar内bid_amount10序列的差分，每个值用当前tick的bid_amount10来nomalize，再求kurt | bar内bid_amount10序列的差分，每个值除以当前tick的bid_amount10，丢弃因分母为0而算出的非有限值，再求kurt | 求kurt时若样本数<4，此字段填nan |
| 119 | book10_ratio_avg_from_tick | float64 | bar内每个tick求book10_ratio，然后求avg | 丢弃无法计算book10_ratio的tick，剩下的每个tick求book10_ratio，得到的序列求avg(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求avg时样本集为空集，则此字段值为nan |
| 120 | book10_ratio_std_from_tick | float64 | bar内每个tick求book10_ratio，然后求std | 丢弃无法计算book10_ratio的tick，剩下的每个tick求book10_ratio，得到的序列求std(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求std时样本数<2，则此字段值为nan |
| 121 | book10_ratio_skew_from_tick | float64 | bar内每个tick求book10_ratio，然后求skew | 丢弃无法计算book10_ratio的tick，剩下的每个tick求book10_ratio，得到的序列求skew(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求skew时样本数<3，则此字段值为nan |
| 122 | book10_ratio_kurt_from_tick | float64 | bar内每个tick求book10_ratio，然后求kurt | 丢弃无法计算book10_ratio的tick，剩下的每个tick求book10_ratio，得到的序列求kurt(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求kurt时样本数<4，则此字段值为nan |
| 123 | book10_ratio_chg_avg_from_tick | float64 | bar内每个tick求book10_ratio，然后做差分，差分序列求avg | 去掉无法计算book10_ratio的tick之后，剩下的tick求book10_ratio，然后做差分，得到的序列再求avg(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求avg时样本集为空集，则此字段值为nan |
| 124 | book10_ratio_chg_std_from_tick | float64 | bar内每个tick求book10_ratio，然后做差分，差分序列求std | 去掉无法计算book10_ratio的tick之后，剩下的tick求book10_ratio，然后做差分，得到的序列再求std(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求std时样本数<2，则此字段值为nan |
| 125 | book10_ratio_chg_skew_from_tick | float64 | bar内每个tick求book10_ratio，然后做差分，差分序列求skew | 去掉无法计算book10_ratio的tick之后，剩下的tick求book10_ratio，然后做差分，得到的序列再求skew(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求skew时样本数<3，则此字段值为nan |
| 126 | book10_ratio_chg_kurt_from_tick | float64 | bar内每个tick求book10_ratio，然后做差分，差分序列求kurt | 去掉无法计算book10_ratio的tick之后，剩下的tick求book10_ratio，然后做差分，得到的序列再求kurt(book10_ratio的定义见2.3.3.7 $book\_ratio$(#2337-book_ratio)) | 如果求kurt时样本数<4，则此字段值为nan |
| 127 | book10_rratio_avg_from_tick | float64 | bar内每个tick求book10_rratio，然后求avg | 丢弃无法计算book10_rratio的tick，剩下的每个tick求book10_rratio，得到的序列求avg(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求avg时样本集为空集，则此字段值为nan |
| 128 | book10_rratio_std_from_tick | float64 | bar内每个tick求book10_rratio，然后求std | 丢弃无法计算book10_rratio的tick，剩下的每个tick求book10_rratio，得到的序列求std(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求std时样本数<2，则此字段值为nan |
| 129 | book10_rratio_skew_from_tick | float64 | bar内每个tick求book10_rratio，然后求skew | 丢弃无法计算book10_rratio的tick，剩下的每个tick求book10_rratio，得到的序列求skew(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求skew时样本数<3，则此字段值为nan |
| 130 | book10_rratio_kurt_from_tick | float64 | bar内每个tick求book10_rratio，然后求kurt | 丢弃无法计算book10_rratio的tick，剩下的每个tick求book10_rratio，得到的序列求kurt(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求kurt时样本数<4，则此字段值为nan |
| 131 | book10_rratio_chg_avg_from_tick | float64 | bar内每个tick求book10_rratio，然后做差分，差分序列求avg | 去掉无法计算book10_rratio的tick之后，剩下的tick求book10_rratio，然后做差分，得到的序列再求avg(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求avg时样本集为空集，则此字段值为nan |
| 132 | book10_rratio_chg_std_from_tick | float64 | bar内每个tick求book10_rratio，然后做差分，差分序列求std | 去掉无法计算book10_rratio的tick之后，剩下的tick求book10_rratio，然后做差分，得到的序列再求std(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求std时样本数<2，则此字段值为nan |
| 133 | book10_rratio_chg_skew_from_tick | float64 | bar内每个tick求book10_rratio，然后做差分，差分序列求skew | 去掉无法计算book10_rratio的tick之后，剩下的tick求book10_rratio，然后做差分，得到的序列再求skew(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求skew时样本数<3，则此字段值为nan |
| 134 | book10_rratio_chg_kurt_from_tick | float64 | bar内每个tick求book10_rratio，然后做差分，差分序列求kurt | 去掉无法计算book10_rratio的tick之后，剩下的tick求book10_rratio，然后做差分，得到的序列再求kurt(book10_rratio的定义见2.3.3.8 $book\_rratio$(#2338-book_rratio)) | 如果求kurt时样本数<4，则此字段值为nan |
| 135 | twap_from_tick | float64 | bar内收到的所有tick的last_price的均值 | | |
| 136 | arrival_time_from_trans | datetime | 基于逐笔的字段计算完毕时的时间 | 基于逐笔的字段进入最后一步计算时(即用已经记录的中间结果生成分钟bar)，此时间戳被生成，然后计算任务进入任务队列，过一段时间任务出队时就会开始计算。更多细节请参照`README.md`中的架构介绍。 | |
| 137 | total_trades_from_trans | int64 | bar内收到的成交笔数 | | |
| 138 | twap_from_trans | float64 | bar内成交价格的算术均值 | 记录bar内所有成交的成交价，求算术均值。 | 如果bar内未收到逐笔成交，则向前找最新一笔成交记录的成交价格。如果今天还未发生过成交，则此字段填nan。 |
| 139 | buy_amount_by_bsflag_from_trans | float64 | 使用`by_bsflag`规则算出的bar内买额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |
| 140 | sell_amount_by_bsflag_from_trans | float64 | 使用`by_bsflag`规则算出的bar内卖额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |

| 序号 | 字段名 | 类型 | 释义 | 计算方法 | 特殊处理 |
| --- | --- | --- | --- | --- | --- |
| 141 | buy_amount_by_tick_from_trans | float64 | 使用by_tick规则算出的bar内买额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |
| 142 | sell_amount_by_tick_from_trans | float64 | 使用by_tick规则算出的bar内卖额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |
| 143 | buy_amount_by_quote_from_trans | float64 | 使用by_quote规则算出的bar内买额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |
| 144 | sell_amount_by_quote_from_trans | float64 | 使用by_quote规则算出的bar内卖额 | 参考2.3.3.9 $buy_amount$与$sell_amount$(#2339-buy_amount与sell_amount) | |

### 2.3.2. 基于逐笔的字段

### 2.3.3. 统计量定义

#### 2.3.3.1. *MID_PRICE*

对于一个tick，定义其$mid_price$如下:

- 若$ask_1$和$bid_1$皆存在，则$mid_price = (ask_price_1 + bid_price_1)/2$
- 若$ask_1$存在但$bid_1$不存在，则$mid_price = ask_price_1$
- 若$ask_1$不存在但$bid_1$存在，则$mid_price = bid_price_1$
- 若$ask_1$和$bid_1$皆不存在，则认为$mid_price$无法计算

#### 2.3.3.2. *SPREAD*

对于一个tick，定义其$spread$如下:

- 如果$ask_1$和$bid_1$皆存在，则$spread = (ask_price_1 - bid_price_1)/mid$
- 如果$ask_1$和$bid_1$中任何一个不存在，则认为$spread$无法计算

#### 2.3.3.3 *DELTA_AMOUNT_ASK*

$delta_amount_ask$约定了四种算法，由四种算法算出的字段以后缀$_algo_i$区分。一个分钟的$delta_amount_ask$是这分钟内所有tick的$delta_amount_ask$之和。对于一个tick，其$delta_amount_ask$的计算方法在下文介绍:

四种算法的区别如下表格所示:
| |只考虑第一档|考虑所有档|
|--|--|--|
|不考虑负变化量|$algo_1$|$algo_2$|
|考虑负变化量|$algo_3$|$algo_4$|

记$A_i^{this}, P_i^{this}$分别为当前tick的卖$i$档price与amount。记$A_i^{last}, P_i^{last}$分别为前一个tick的卖$i$档price与amount。

如果此tick和前一个tick都没有$ask_1$，则对于此tick，

$$
\begin{aligned}
delta\_amount\_ask\_algo_1 &= 0 \\
delta\_amount\_ask\_algo_2 &= 0 \\
delta\_amount\_ask\_algo_3 &= 0 \\
delta\_amount\_ask\_algo_4 &= 0
\end{aligned}
$$

$delta\_amount\_ask\_algo_1 = delta\_amount\_ask\_algo_2 = delta\_amount\_ask\_algo_3 = delta\_amount\_ask\_algo_4 = 0$。

如果此tick有$ask_1$，且上一个tick没有$ask_1$，则

$$
\begin{aligned}
delta\_amount\_ask\_algo_1 &= A_1^{this} \\
delta\_amount\_ask\_algo_2 &= \sum_i A_i^{this} \\
delta\_amount\_ask\_algo_3 &= A_1^{this} \\
delta\_amount\_ask\_algo_4 &= \sum_i A_i^{this}
\end{aligned}
$$

如果此tick没有$ask_1$，且上一个tick有$ask1$，则

$$
delta\_amount\_ask\_algo_1 = 0 delta\_amount\_ask\_algo_2 = 0 delta\_amount\_ask\_algo_3 = -A_1^{last} delta\_amount\_ask\_algo_4 = -\sum_i A_i^{last}
$$

如果此tick和上一个tick都有$ask1$，则分三种情况:

当$P_1^{this} > P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_ask\_algo_1 &= 0 \\
delta\_amount\_ask\_algo_2 &= 0 \\
delta\_amount\_ask\_algo_3 &= -A_1^{last} \\
delta\_amount\_ask\_algo_4 &= -\sum_i A_i^{last}[P_1^{this} > P_i^{last}] + \sum_i (A_1^{this} - A_i^{last})[P_1^{this} = P_i^{last}]
\end{aligned}
$$

(注意公式中的$[condition]$记号，当$condition$成立时$[condition] = 1$，否则$[condition] = 0$)

当$P_1^{this} < P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_ask\_algo_1 &= A_1^{this} \\
delta\_amount\_ask\_algo_2 &= \sum_i A_i^{this}[P_i^{this} < P_1^{last}] + \sum_i (A_i^{this} - A_1^{last})[P_i^{this} = P_1^{last}] \\
delta\_amount\_ask\_algo_3 &= A_1^{this} \\
delta\_amount\_ask\_algo_4 &= \sum_i A_i^{this}[P_i^{this} < P_1^{last}] + \sum_i (A_i^{this} - A_1^{last})[P_i^{this} = P_1^{last}]
\end{aligned}
$$

当$P_1^{this} = P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_ask\_algo_1 &= A_1^{this} - A_1^{last} \\
delta\_amount\_ask\_algo_2 &= A_1^{this} - A_1^{last} \\
delta\_amount\_ask\_algo_3 &= A_1^{this} - A_1^{last} \\
delta\_amount\_ask\_algo_4 &= A_1^{this} - A_1^{last}
\end{aligned}
$$

#### 2.3.3.4 *DELTA_AMOUNT_BID*

$delta_amount_bid$约定了四种算法，由四种算法算出的字段以后缀$_algo_i$区分。一个分钟的$delta_amount_bid$是这分钟内所有tick的$delta_amount_bid$之和。对于一个tick，其$delta_amount_bid$的计算方法在下文介绍：

四种算法的区别如下表格所示：

| | **只考虑第一档** | **考虑所有档** |
|---|---|---|
| 不考虑负变化量 | $algo_1$ | $algo_2$ |
| 考虑负变化量 | $algo_3$ | $algo_4$ |

记$A_i^{this}, P_i^{this}$分别为当前tick的买$i$档price与amount。记$A_i^{last}, P_i^{last}$分别为前一个tick的买$i$档price与amount。

如果此tick和前一个tick都没有$bid_1$，则对于此tick，

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= 0 \\
delta\_amount\_bid\_alog_2 &= 0 \\
delta\_amount\_bid\_alog_3 &= 0 \\
delta\_amount\_bid\_alog_4 &= 0
\end{aligned}
$$

$delta\_amount\_bid\_alog_1 = delta\_amount\_bid\_alog_2 = delta\_amount\_bid\_alog_3 = delta\_amount\_bid\_alog_4 = 0$。

如果此tick有$bid_1$，且上一个tick没有$bid_1$，则

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= A_1^{this} \\
delta\_amount\_bid\_alog_2 &= \sum_i A_i^{this} \\
delta\_amount\_bid\_alog_3 &= A_1^{this} \\
delta\_amount\_bid\_alog_4 &= \sum_i A_i^{this}
\end{aligned}
$$

如果此tick没有$bid_1$，且上一个tick有$bid1$，则

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= 0 \\
delta\_amount\_bid\_alog_2 &= 0 \\
delta\_amount\_bid\_alog_3 &= -A_1^{last} \\
delta\_amount\_bid\_alog_4 &= -\sum_i A_i^{last}
\end{aligned}
$$

如果此tick和上一个tick都有$bid1$，则分三种情况：

当$P_1^{this} > P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= A_1^{this} \\
delta\_amount\_bid\_alog_2 &= \sum_i A_i^{this}[P_i^{this} > P_1^{last}] + \sum_i (A_i^{this} - A_1^{last})[P_i^{this} = P_1^{last}] \\
delta\_amount\_bid\_alog_3 &= A_1^{this} \\
delta\_amount\_bid\_alog_4 &= \sum_i A_i^{this}[P_1^{this} > P_1^{last}] + \sum_i (A_i^{this} - A_1^{last})[P_i^{this} = P_1^{last}]
\end{aligned}
$$

(注意公式中的$[condition]$记号，当$condition$成立时$[condition] = 1$，否则$[condition] = 0$)

当$P_1^{this} < P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= 0 \\
delta\_amount\_bid\_alog_2 &= 0 \\
delta\_amount\_bid\_alog_3 &= -A_1^{last} \\
delta\_amount\_bid\_alog_4 &= -\sum_i A_i^{last}[P_1^{this} < P_i^{last}] + \sum_i (A_i^{last} - A_1^{this})[P_1^{this} = P_i^{last}]
\end{aligned}
$$

当$P_1^{this} = P_1^{last}$时，

$$
\begin{aligned}
delta\_amount\_bid\_alog_1 &= A_1^{this} - A_1^{last} \\
delta\_amount\_bid\_alog_2 &= A_1^{this} - A_1^{last} \\
delta\_amount\_bid\_alog_3 &= A_1^{this} - A_1^{last} \\
delta\_amount\_bid\_alog_4 &= A_1^{this} - A_1^{last}
\end{aligned}
$$

#### 2.3.3.5 QIMB

对于一个tick，其$qimb$定义如下：

$qimb = \frac{ask\_amount - bid\_amount}{ask\_amount + bid\_amount}$

qimbn是指用委托簿前n档买和前n档卖来计算ask_amount和bid_amount。

#### 2.3.3.6 TICK_RETURN

对于一个tick，其$tick_return$定义如下：

$tick\_return = \frac{mid\_price}{last\_mid\_price}$

其中$mid_price$是此tick的mid_price。

向前追溯到最近的一个能计算mid_price的tick，将那个tick的mid_price作为last_mid_price。

如果当前tick的mid_price无法计算，或往前追溯找不到可计算mid_price的tick，就认为此tick的tick_return无法计算。

#### 2.3.3.7 BOOK_RATIO

对于一个tick，其$book_ratio$定义为

$book\_ratio = \frac{ask\_amount}{bid\_amount}$

当分母为0时，认为此tick的$book_ratio$无法计算。

book10_ratio指使用委托簿卖前十档和买前十档来求$ask_amount$和$bid_amount$

#### 2.3.3.8 BOOK_RRATIO

对于一个tick，其$book_rratio$定义为

$book\_rratio = \frac{买盘前5档委托额/买盘后五档委托额}{卖盘前5档委托额/卖盘后五档委托额}$

如果"买盘后五档委托额"、"卖盘前5档委托额"、"卖盘后五档委托额"中任何一项等于0，就认为此tick的$book_rratio$无法计算。

#### 2.3.3.9 $BUY_AMOUNT$与$SELL_AMOUNT$

buy_amount和sell_amount分别表示一个分钟内的主动买量和主动卖量。目前有如下几个字段：

$$buy_amount_by_bsflag, sell_amount_by_bsflagbuy_amount_by_tick, sell_amount_by_tickbuy_amount_by_quote, sell_amount_by_quote$$

其中`by_xxx`表示判断规则，根据具体某种判断规则，每次成交会被判断为主动买、主动卖或半买半卖。如果一次成交被判断为主动买，则此次成交的成交额算进`buy_amount`；如果一次成交被判断为主动卖，则成交额算进`sell_amount`；如果被判定为半买半卖，则成交额的一半算进`buy_amount`、一半算进`sell_amount`。

`by_bsflag`规则：

- 集合竞价期间的成交判定为半买半卖，连续竞价成交使用下述判断规则
- 如果买方的订单号更大，则认为是主动买
- 如果卖方的订单号更大，则认为是主动卖

`by_tick`规则：

- 集合竞价期间的成交判定为半买半卖，连续竞价成交使用下述判断规则
- 如果当前成交价比上一笔成交价格高，则认为是主动买
- 如果当前成交价比上一笔成交价低，则认为是主动卖
- 如果当前成交价等于上一笔成交价，则这笔成交的方向与上一笔成交根据`by_tick`判断出的方向相同
- 如果当前成交价等于上一笔成交价，且不存在上一笔成交(此次成交为第一笔)，则判定为半买半卖

`by_quote`规则：

- 集合竞价期间的成交判定为半买半卖，连续竞价成交使用下述判断规则
- 找到目前收到的这只股票的最新tick，计算mid_price，如果mid_price不可计算，则使用`by_tick`规则的判定方向作为`by_quote`的判定方向。
- 如果mid_price可计算，当前成交价高于mid_price则认为是主动买，当前成交价低于mid_price则认为是主动卖，当前价格等于mid_price就用`by_tick`判定的方向。

## 3. 后续版本需要解决的问题

### 3.1.加入更多基于逐笔的字段

依据张文彬总的需求文档，实现剩下那些基于逐笔的字段。

### 3.2. 解决15点47分集合竞价时tick的判定问题

为了避免(14:57,14:58]这根bar的成交量大于0，应当把这根bar中开头几秒的tick判定到上一分钟去。

特殊处理上交所加入集合竞价规则之前的日期。

### 3.3. 对没收到tick的分钟设置一个特殊流程

如果某个分钟某支票没收到tick，则进入一个特殊流程，而不是像现在这样补tick。

### 3.4. 盘后真实还原盘中时钟

现在盘中使用定时任务，盘后纯依靠行情时间戳，这样会导致潜在的盘中盘后不一致。

盘后可以想办法真实模拟盘中时钟（比如用优先队列维护事件）。

### 3.5. 每只票收齐则立刻发送分钟线

现在是只有在分钟末才会计算所有分钟线，并批量发出。为了降低时延，考虑每只票一旦收齐就单独发。

### 3.6. 考虑加入盘中拉起的功能

程序中途重启，则先从bdf_server读取之前的数据，计算完之前的分钟线，再转到实时状态。

### 3.7. 优化掉性能瓶颈

- 考虑如何优化zstd解压的过程，看下是否可以并行
- 考虑修改现有的星形架构，星形架构中间的部分已经成为性能瓶颈。