## **vn.py社区精选18 - 老用户福音，MongDB分表重构！**

原创 用Python的交易员 VNPY  2020-02-17

收录于话题

#vn.py社区精选

25个

原文作者：雍木

1.0时代的vn.py只支持MongoDB数据库，在数据结构上使用MongoDB的数据库（DATABASE）来区分不同的历史数据时间周期，使用集合（COLLECTION）来区分不同的交易合约数据，许多用户（尤其以职业交易员背景居多）喜欢这种一目了然的数据结构。

2.0版本在数据库的设计上则是选择同时兼容SQL（SQLite、MySQL、PostgreSQL）和NoSQL（MongoDB）两种不同范式，使用软件工程中ORM（对象关系印射）/ODM（对象文件印射）构建了独立的抽象中间层（vnpy.trader.database），在数据存储的结构上发生了比较大的变化。

以2.0版本中MongoDB数据库为例，首先需要创建一个单独的数据库，通常命名为vnpy。在该数据库中只包含两张表，分别用于存放K线和tick数据：

db\_bar\_data集合

存放所有K线数据，包括不同的交易合约以及时间周期

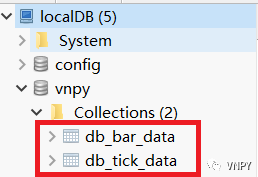
通过vt\_symbol+interval+datetime三个字段来保证唯一性

db\_tick\_data集合

存放所有Tick数据

通过vt\_symbol+datetime两个字段来保证唯一性

通过Robo3T客户端查看的结构如下图：



尽管这种设计保证了和SQL类数据库的兼容性，但缺陷是随着使用时间的增长，集合中数据的长度和宽度都在增加。刚开始数据库可能只存5年内IF88的数据，到后来变成了所有国内期货品种从开始上市至今的所有1分钟历史数据，此时单一集合的数据总量会变得相当大，读取数据时查询检索很费时。

本文中试着给出一种解决方案：

若指定特定的集合名，则保存在该集合中；

若没有指定，则将数据仍保存在原有的集合中。

**配置数据库为MongDB**

如果数据库已经配置为MongoDB可以直接跳过这一段。找到用户目录下.vntrader文件夹，用VSCode打开vt\_setting.json文件，对全局配置中的数据库相关字段进行以下修改。这里我们为了演示方便，创建一个新的名为【vnpytest】的数据库：

"database.driver": "mongodb","database.database": "vnpytest","database.host": "localhost","database.port": 27017,"database.user": "","database.password": "","database.authentication\_source": ""

如果尚未安装MongDB数据库以及可视化工具，可以参考往期文章：[vn.py社区精选6 - 做交易，你需要选好数据库](http://mp.weixin.qq.com/s?__biz=MzI1MTQ2Njc5OQ==&mid=2247483949&idx=1&sn=55a038e9b6729196dbcf13439418df73&chksm=e9f3c76dde844e7b9143684c1ef936082706ed6de29dada62d8aa39b09992d753222cbe77340&scene=21" \l "wechat_redirect" \t "/Users/huangning/Documents\\x/_blank)。

**修改database.py文件**

找到vn.py源代码所在的路径，进入目录vnpy/trader/database打开database.py 文件，需要对save\_bar\_data和save\_tick\_data这两个函数进行修改，主要增加了一个参数collection\_name：

@abstractmethoddef save\_bar\_data( self, datas: Sequence["BarData"], collection\_name: str = None,): pass  
@abstractmethoddef save\_tick\_data( self, datas: Sequence["TickData"], collection\_name: str = None,): pass

**修改database\_mongo.py文件**

然后打开位于同一目录下的database\_mongo.py 文件，同样我们需要对 save\_bar\_data 和 save\_tick\_data进行修改。

改动后的数据写入逻辑如下：

若没有指定collection\_name，数据存入db\_bar\_data或db\_tick\_data；

若指定了collection\_name，则调用switch\_collection函数，把数据切换到写入到指定的Collection中。

from mongoengine.context\_managers import switch\_collection def save\_bar\_data(self, datas: Sequence[BarData], collection\_name: str = None): for d in datas: updates = self.to\_update\_param(d) updates.pop("set\_\_gateway\_name") updates.pop("set\_\_vt\_symbol") if collection\_name is None: ( DbBarData.objects( symbol=d.symbol, interval=d.interval.value, datetime=d.datetime ).update\_one(upsert=True, \*\*updates) ) else: with switch\_collection(DbBarData, collection\_name): ( DbBarData.objects( symbol=d.symbol, interval=d.interval.value, datetime=d.datetime ).update\_one(upsert=True, \*\*updates) )  
 def save\_tick\_data(self, datas: Sequence[TickData], collection\_name: str = None): for d in datas: updates = self.to\_update\_param(d) updates.pop("set\_\_gateway\_name") updates.pop("set\_\_vt\_symbol") if collection\_name is None: ( DbTickData.objects( symbol=d.symbol, exchange=d.exchange.value, datetime=d.datetime ).update\_one(upsert=True, \*\*updates) ) else: with switch\_collection(DbTickData, collection\_name): ( DbTickData.objects( symbol=d.symbol, exchange=d.exchange.value, datetime=d.datetime ).update\_one(upsert=True, \*\*updates) )

**Jupyter Notebook使用示例**

最后可以通过Jupyter Notebook来测试下使用效果，这里我们将把CSV格式的1小时周期的XBTUSD数据，载入到指定的集合【XBTUSD】中。

具体步骤如下：

调用csv库读取"XBTUSD.csv"文件到内存中，其中DictReader类似于常规reader，但是将每行中的信息映射到一个字典，该字典的key由表头可选参数给出；

遍历每行的rows对象，生成BarData对象，并且缓存至bars列表中；

调用save\_bar\_data函数，把bars列表中的数据写入到我们指定的集合【XBTUSD】中。

注意：若该函数第二个参数collection\_name为空，则bars列表中的数据会被写入默认的数据集合【db\_bar\_data】中。

from vnpy.trader.database import database\_managerfrom vnpy.trader.object import BarDatafrom vnpy.trader.constant import Interval, Exchangeimport csv  
file\_name='XBTUSD.csv'with open(file\_name,newline='',encoding='UTF-8') as csvfile: rows=csv.DictReader(csvfile) bars = [] for i in rows: bar = BarData( symbol=i["symbol"], exchange=Exchange.BITMEX, datetime=i["datetime"], interval=Interval.HOUR, volume=i["volume"], open\_price=i["open"], high\_price=i["high"], low\_price=i["low"], close\_price=i["close"], gateway\_name='DB', ) bars.append(bar) print('Saving data to database ...') database\_manager.save\_bar\_data(bars, "XBTUSD")

载入完成后，打开MongDB客户端Robo3T，我们可以看到在vnpytest数据库下新增了一个集合【XBTUSD】，且所有1小时K线数据均已成功写入其中：

