

38 | MySQL：日志和数据存储系统

2019-08-05 景霄

Python核心技术与实战

[进入课程 >](#)



讲述：冯永吉

时长 12:54 大小 11.83M



你好，我是景霄。今天这节课，我们来聊聊日志和存储系统。

在互联网公司中，日志系统是一个非常重要的技术底层。在每一次重要的交互行为中，关键信息都会被记录下来存档，以供日后线下分析，或者线上实时分析。这些数据，甚至可以说是硅谷互联网大公司的命脉所在。

有了它们，你才能建立机器学习模型来预测用户的行为，从而可以精确描绘用户画像，然后针对性地使用推荐系统、分类器，将用户进一步留下，并精准推送广告来盈利。

在量化交易中，日志同样有着非常重要的作用。一如前面所讲，我们重要的数据有：行情数据、策略信号、执行情况、仓位信息等等非常多的信息。

对于简单的、小规模的数据，例如 orderbook 信息，我们完全可以把数据存在 txt、csv 文件中，这样做简单高效。不过，缺点是，随着数据量上升，一个文件将会变得非常大，检索起来也不容易。这时，一个很直观的方式出现了，我们可以把每天的数据存在一个文件中，这样就暂时缓解了尴尬。

但是，随着数据量的上升，或者是你的算法逐渐来到高频交易领域时，简单地把数据存在文件上，已经不足以满足新的需求，更无法应对分布式量化交易系统的需求。于是，一个显而易见的想法就是，我们可以把日志存在数据库系统中。

这节课，我们就以 MySQL 这种传统型关系数据库为例，讲解一下数据库在日志中的运用。

快速理解 MySQL

担心一些同学没有数据库的基础，我先来简单介绍一下 MySQL 数据库。

MySQL 属于典型的关系型数据库（RDBMS），所谓的关系型数据库，就是指建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法，来处理数据库中的数据。基本上任何学习资料都会告诉你，它有着下面这几个特征：

1. 数据是以表格的形式出现的；
2. 每一行是各种记录名称；
3. 每一列是记录名称所对应的数据域；
4. 许多的行和列，组成一张表单；
5. 若干的表单，组成数据库（database）这个整体。

不过，抛开这些抽象的特征不谈，你首先需要掌握的，是下面这些术语的概念。

数据库，是一些关联表的集合；而数据表则是数据的矩阵。在一个数据库中，数据表看起来就像是一个简单的电子表格。

在数据表中，每一列包含的是相同类型的数据；每一行则是一组相关的数据。

主键也是数据表中的一个列，只不过，这一列的每行元素都是唯一的，且一个数据表中只能包含一个主键；而外键则用于关联两个表。

除此之外，你还需要了解索引。索引是对数据库表中一列或多列的值进行排序的一种结构。使用索引，我们可以快速访问数据库表中的特定信息。一般来说，你可以对很多列设置索引，这样在检索指定列的时候，就大大加快了速度，当然，代价是插入数据会变得更慢。

至于操作 MySQL，一般用的是结构化查询语言 SQL。SQL 是一种典型的领域专用语言（domain-specific language，简称 DSL），这里我就不做过多介绍了，如果你感兴趣，可以学习极客时间平台上的“[SQL 必知必会](#)”专栏。

接下来，我们就来简单看一下，如何使用 Python 来操作 MySQL 数据库。


Python 连接数据库的方式有好多种，这里我简单介绍其中两种。我们以 Ubuntu 为例，假设你的系统中已经安装过 MySQL Server。（安装 MySQL 可以参考这篇文章 <https://www.jianshu.com/p/3111290b87f4>，或者你可以自行搜索解决）

mysqlclient

事实上，Python 连接 MySQL 最流行的一个驱动是 MySQL-python，又叫 MySQLdb，很多框架都也是基于此库进行开发。不过，遗憾的是，它只支持 Python2.x，而且安装的时候有很多前置条件。因为它是基于 C 开发的库，在 Windows 平台安装非常不友好，经常出现失败的情况。所以，现在我们基本不再推荐使用，取代者是它的衍生版本——mysqlclient。


mysqlclient 完全兼容 MySQLdb，同时支持 Python3.x，是 Django ORM 的依赖工具。如果你想使用原生 SQL 来操作数据库，那么我优先推荐使用这个框架。

它的安装方式很简单：

 复制代码

```
1 sudo apt-get install python3-dev
2 pip install mysqlclient
```

我们来看一个样例代码：

 复制代码

```
1 import MySQLdb
```

```

2
3
4 def test_pymysql():
5     conn = MySQLdb.connect(
6         host='localhost',
7         port=3306,
8         user='your_username',
9         passwd=your_password',
10        db='mysql'
11    )
12
13    cur = conn.cursor()
14    cur.execute('''
15        CREATE TABLE price (
16            timestamp TIMESTAMP NOT NULL,
17            BTCUSD FLOAT(8,2),
18            PRIMARY KEY (timestamp)
19        );
20    ''')
21    cur.execute('''
22        INSERT INTO price VALUES(
23            "2019-07-14 14:12:17",
24            11234.56
25        );
26    ''')
27
28    conn.commit()
29    conn.close()
30
31
32 test_pymy

```

代码的思路很清晰明了，首先是通过 connect 命令连接数据库，来创建一个连接；之后，通过 conn.cursor() 函数创建一个游标。这里你可能会问，为什么要使用游标呢？

一个主要的原因就是，这样可以把集合操作转换成单个记录处理的方式。如果用 SQL 语言从数据库中检索数据，结果会放在内存的一块区域中，并且这个结果往往是一个含有多个记录的集合。而游标机制，则允许用户在 MySQL 内逐行地访问这些记录，这样你就可以按照自己的意愿，来显示和处理这些记录。

继续回到代码中，再往下走，我们创建了一个 price table，同时向里面插入一条 orderbook 数据。这里为了简化代码突出重点，我只保留了 timestamp 和 price。

最后，我们使用 `conn.commit()` 来提交更改，然后 `close()` 掉连接就可以了。

peewee


不过，大家逐渐发现，写原生的 SQL 命令很麻烦。因为你需要根据特定的业务逻辑，来构造特定的插入和查询语句，这可以说就完全抛弃了面向对象的思维。因此，又诞生了很多封装 wrapper 包和 ORM 框架。

这里所说的 ORM（Object Relational Mapping，简称 ORM），是 Python 对象与数据库关系表的一种映射关系，有了 ORM 后，我们就不再需要写 SQL 语句，而可以直接使用 Python 的数据结构了。

ORM 框架的优点，是提高了写代码的速度，同时兼容多种数据库系统，如 SQLite、MySQL、PostgreSQL 等这些数据库；而付出的代价，可能就是性能上的一些损失。


接下来要讲的 peewee，正是其中一种基于 Python 的 ORM 框架，它的学习成本非常低，可以说是 Python 中最流行的 ORM 框架。

它的安装方式也很简单：

 复制代码

```
1 pip install peewee
```

我们来看一个样例代码：

 复制代码

```
1 import peewee
2 from peewee import *
3
4 db = MySQLDatabase('mysql', user='your_username', passwd=your_password')
5
6
7 class Price(peewee.Model):
8     timestamp = peewee.DateTimeField(primary_key=True)
9     BTCUSD = peewee.FloatField()
10
11     class Meta:
12         database = db
```

```
13
14
15 def test_peewee():
16     Price.create_table()
17     price = Price(timestamp='2019-06-07 13:17:18', BTCUSD='12345.67')
18     price.save()
19
20
21 test_p
```

如果你写过 Django，你会发现，这个写法和 Django 简直一摸一样。我们通过一个 Python class，映射了 MySQL 中的一张数据表；只要对其中每一列数据格式进行定义，便可按照 Python 的方式进行操作。

显而易见，peewee 的最大优点，就是让 SQL 语言瞬间变成强类型语言，这样不仅极大地增强了可读性，也能有效减少出 bug 的概率。


不过，事实上，作为一名数据科学家，或者作为一名量化从业者（quant），你要处理的数据远比这些复杂很多。互联网工业界有大量的脏数据，金融行业的信噪比更是非常之低，数据处理只能算是基本功。

如果你对数据分析有兴趣和志向，在学生时期就应该先打牢数学和统计的基础，之后在实习和工作中快速掌握数据处理的方法。当然，如果你已经错过学生时期的话，现在开始也是个不错的选择，毕竟，逐渐形成自己的核心竞争力，才是我们每个人的正道。

量化数据分析系统

数据库有了量化数据存入后，接下来，我们便可以开始进行一些量化分析了。这一块儿也是一个很大的学术领域，叫做时间序列分析，不过就今天这节课的主题来说，我们仅做抛砖引玉，列举一个非常简单的例子，即求过去一个小时 BTC/USD 的最高价和最低价。

我们来看下面这段代码：

 复制代码

```
1 import MySQLdb
2 import numpy as np
3
4
```

```

5 def test_pymysql():
6     conn = MySQLdb.connect(
7         host='localhost',
8         port=3306,
9         user='your_username',
10        passwd='your_password',
11        db='mysql'
12    )
13
14    cur = conn.cursor()
15    cur.execute('''
16        SELECT
17            BTCUSD
18        FROM
19            price
20        WHERE
21            timestamp > now() - interval 60 minute
22    ''')
23
24    BTCUSD = np.array(cur.fetchall())
25    print(BTCUSD.max(), BTCUSD.min())
26
27    conn.close()
28
29
30 test_pym

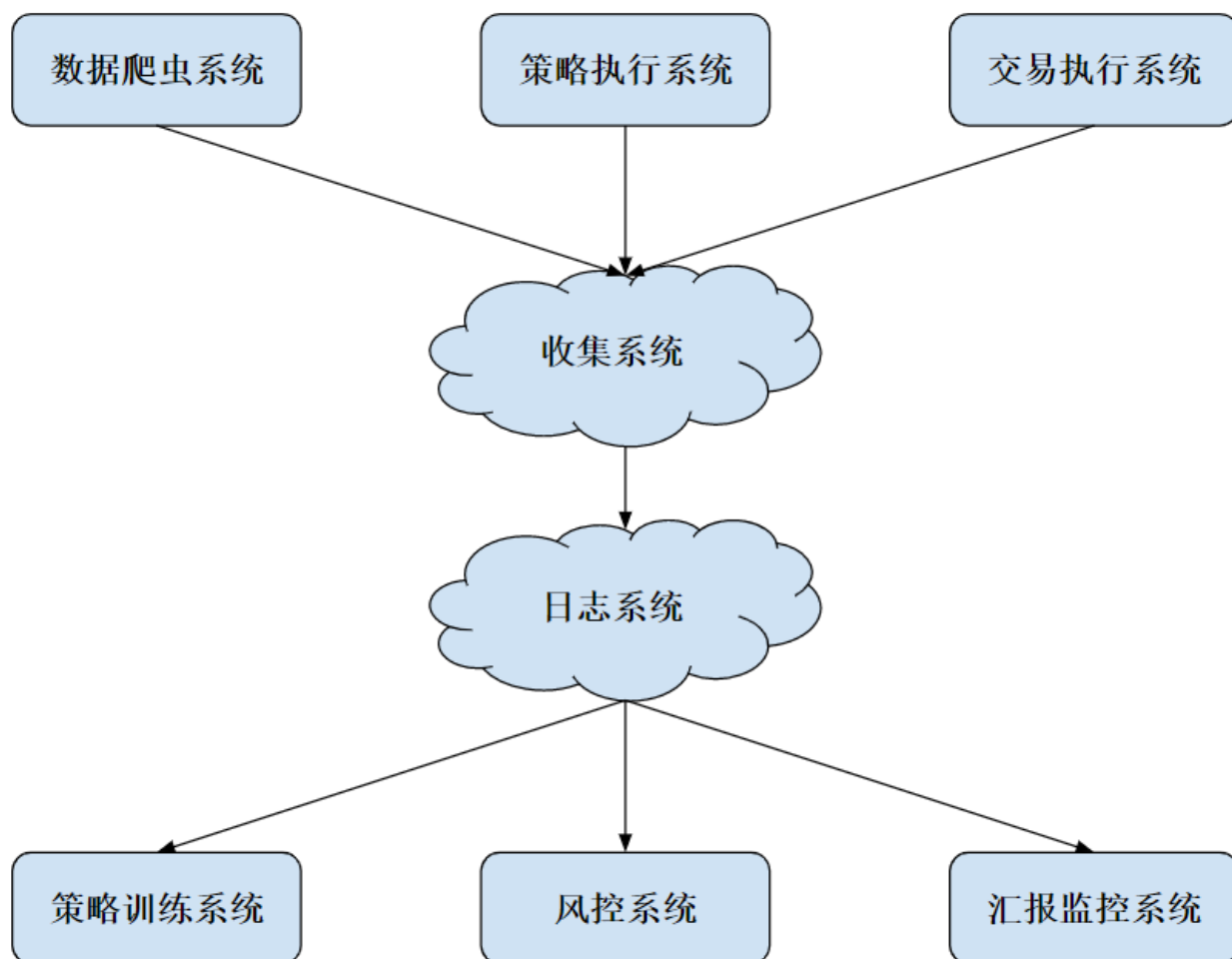
```

代码看起来很简单吧！显然，通过 SQL 语句，我们可以抓取到过去一小时的时间序列片段，拿到我们想要的 BTC/USD 价格向量，然后通过 `numpy` 处理一下即可。不过这里需要注意一点，我们并不需要调用 `conn.commit()`，因为我们的操作是只读的，对数据库没有任何影响。

分布式日志系统

明白了上面的内容后，我们现在来看一下分布式日志系统。

对量化交易而言，我们需要的模块主要有数据系统、策略系统、交易执行系统、线下模型训练、线上风控系统以及实时监控系统。它们之间的对应关系，我画了一张图，你可以参考来理解。



这里的每个子系统都是独立运行的，并且还有许多模块需要迭代更新，所以我们简单保存本地日志显然不是一个明智之举。于是，我们可以专门开一台服务器来运行 MySQL server，并且开放指定端口和其他系统进行交互。

另外，图中的收集系统，其实类似于上一节我们所讲的消息队列体系，在各个上游系统中运行代理工具，负责将各个模块的 log 收集起来，然后发送到收集系统中。收集系统整理过后，再将信息存到日志系统。当然，除了简单的消息队列，我们还能用很多工具，比如阿里云的 Logtail、Apache 的 Flume Agent 等等。

而到了后期，对于日志系统来说，越来越需要注意的就是存储效率和分析效率。随着使用的增加，数据会越来越多，因此我们可以考虑对一些数据进行压缩和保存。而越是久远的数据，越是粗粒度的数据，被调用的概率也就越低，所以它们也就首当其冲，成了我们压缩、保存的目标。

日志分析

最后，我再来补充讲一讲日志的分析。前面提到过，分析一般分为两种，离线分析和在线分析。

在离线分析中，比较常见的是生成报告。

比如，总结某天某月或某季度内的，收益亏损情况（PnL）、最大回撤、夏普比率等数据。这种基于时间窗口的统计，在关系型数据库中也能得到很方便的支持。

而另一类常见的离线使用方式，则是回测系统。在一个新策略研发的周期中，我们需要对历史数据进行回测，这样就可以得到历史数据中交易的收益率等数据。回测系统对于评估一个新的策略非常重要，然而，回测往往需要大量的资源，所以选取好数据库、数据存储方式，优化数据连接和计算，就显得至关重要。

在线分析，则更多应用于风控和警报系统。这种方式，对数据的实时性要求更高一些，于是，一种方法就是，从消息队列中直接拿最快的数据进行操作。当然，这个前提是时间窗口较小，这样你就不需要风控系统来维护大量的本地数据。

至于实时警报，最关键的依然是数据。

比如，数据系统异常停止，被监视的表没有更新；

或者，交易系统的连接出了故障，委托订单的某些状态超过了一定的阈值；

再或者，仓位信息出现了较大的、预计之外的变动。

这些情况都需要进行报警，也就是硅谷大公司所说的“oncall”。一旦发生意外，负责人会迅速收到电话、短信和邮件，然后通过监控平台来确认，是真的出了事故还是监控误报。

当然，现在已经有了不少开源的工具可以在云端使用，其中 AWS 属于全球领先的云计算平台。如果你的服务器架设在美国，那就可以考虑选择它家的各种各样的云服务。这样做的好处是，对于小型量化交易团队而言，避免自己搭建复杂的日志系统，而是把主要精力放在策略的开发迭代之上，提高了不少效率。

总结

这一节课，我从工程的角度，为你介绍了量化系统中的存储系统。我们从基础的 MySQL 的使用方法讲起，再讲到后面的量化系统框架。数据库和数据在绝大部分互联网行业都是核

心，对量化从业者来说也是重要的生产资料。而搭建一套负载合理、数据可靠的数据系统，也需要一个量化团队长期打磨，并根据需求进行迭代。

思考题

最后给你留一道思考题。量化交易需要的数据量不是很大，但是有可能出现调用频率极高的情况，例如回测系统。那么，你能想到哪些优化手段，来降低调用代价吗？欢迎留言和我讨论，也欢迎你把这篇文章分享出去。

 极客时间

Python 核心技术与实战

系统提升你的 Python 能力

景霄

Facebook 资深工程师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 37 | Kafka & ZMQ：自动化交易流水线

精选留言

 写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。