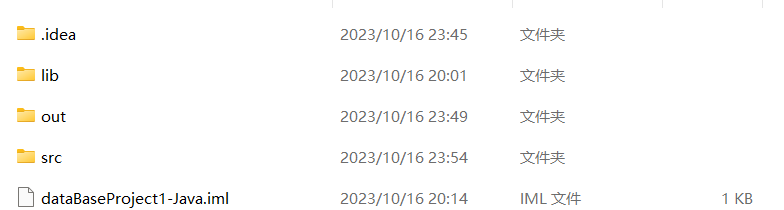
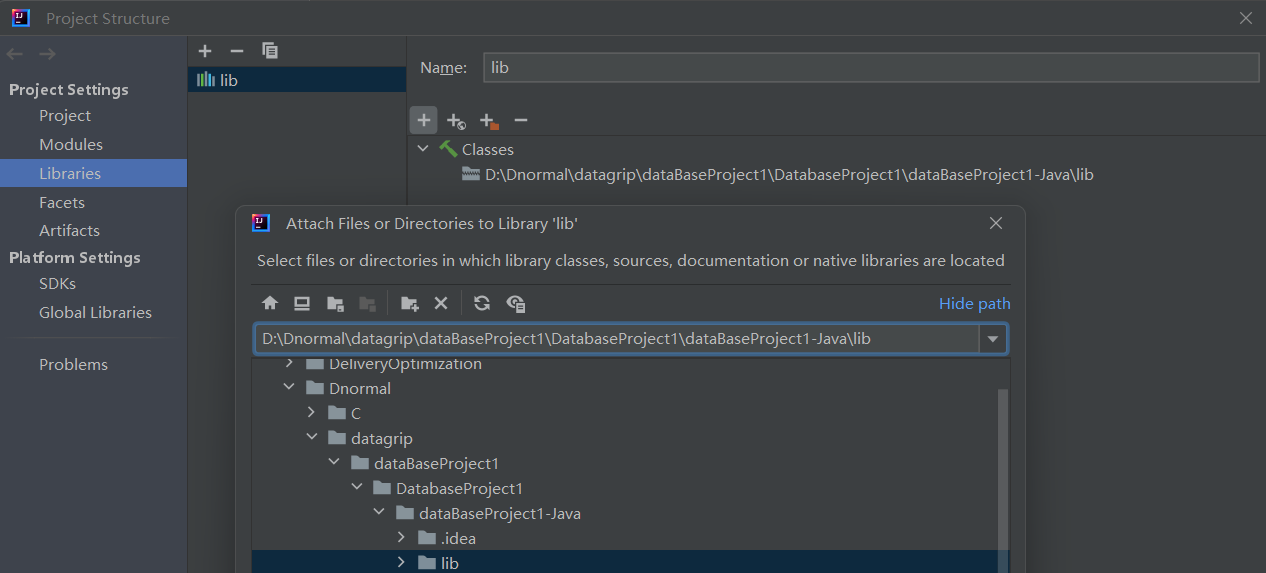
下载连接postgres对应的jar包

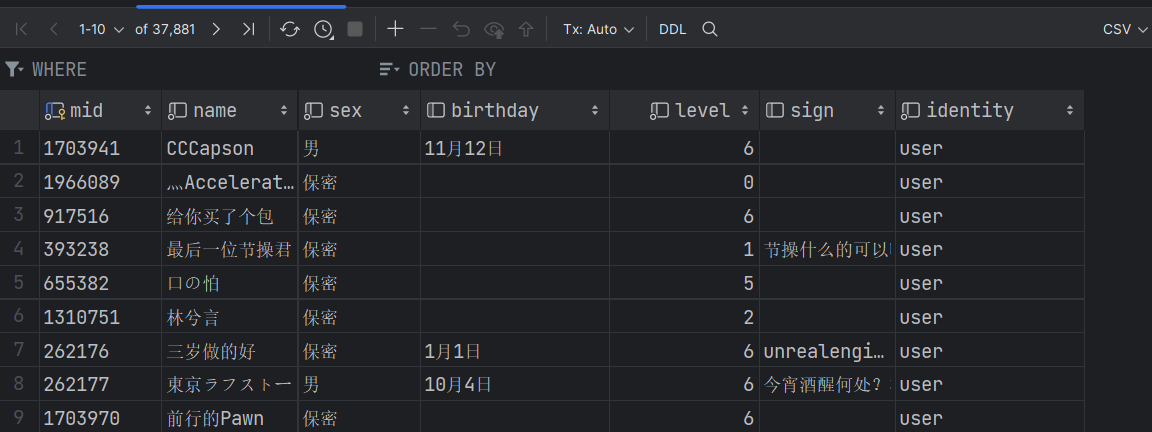


在Java项目目录下新建一个lib文件夹，并将jar包放入



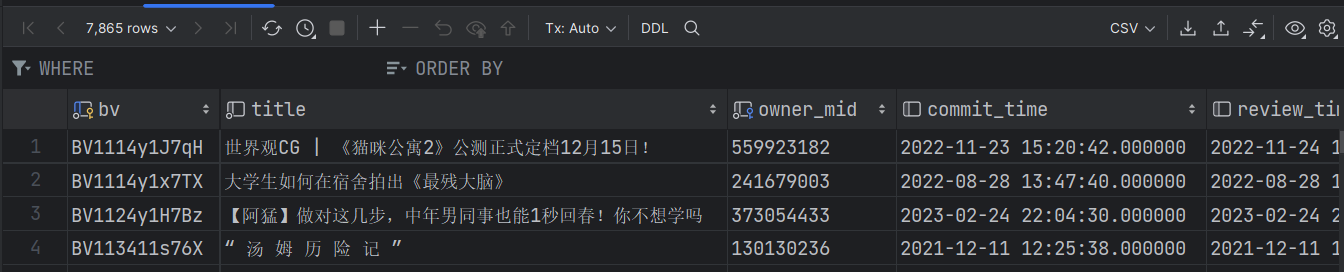
将lib文件夹添加到项目的libraries中

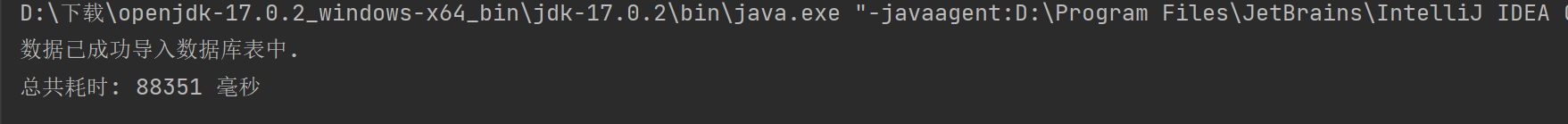


导入user后结果：  


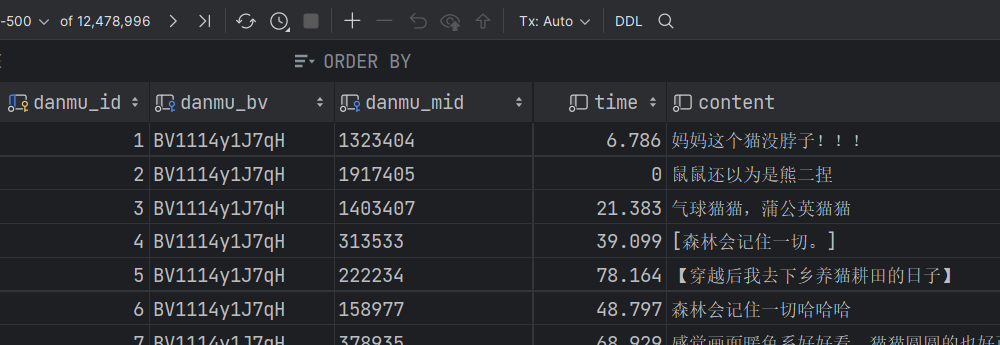


导入video后结果：



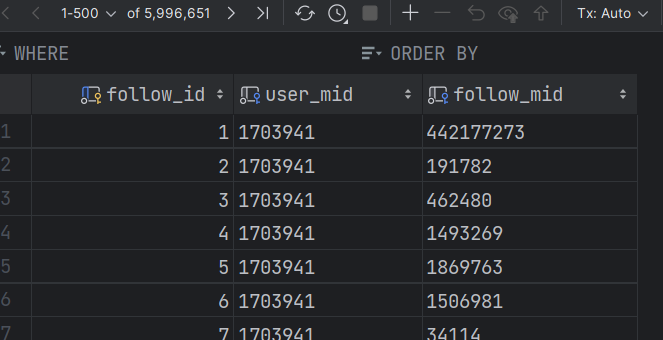


弹幕



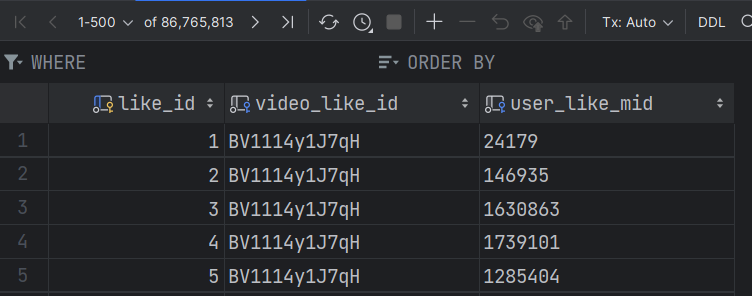


Following



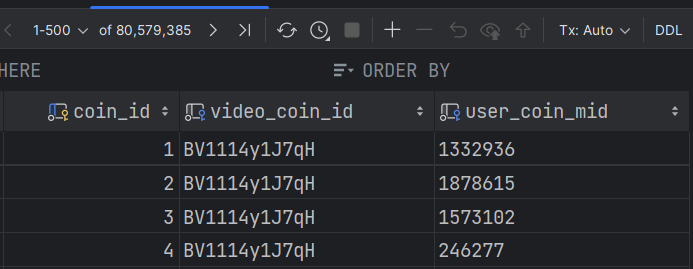


Like relation

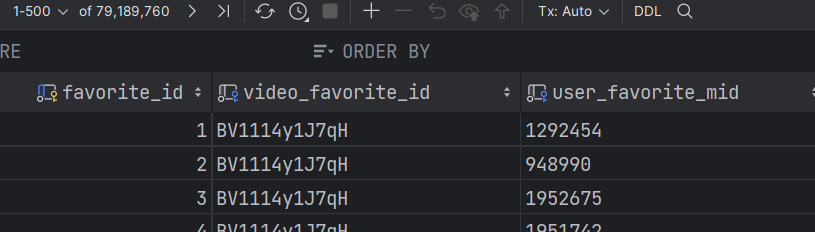


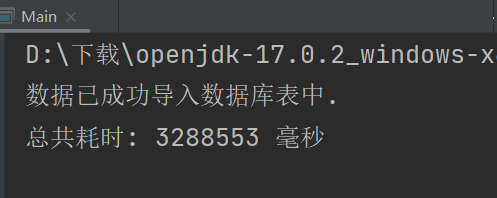


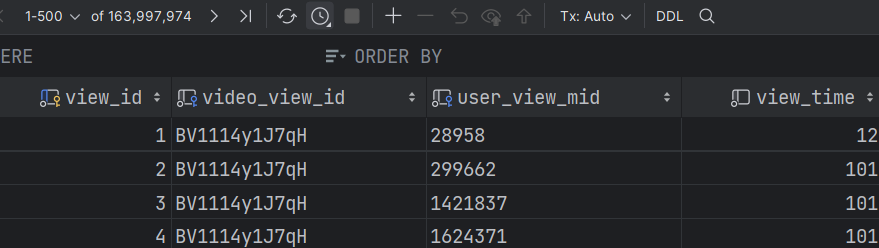
最大堆内存开到10g了



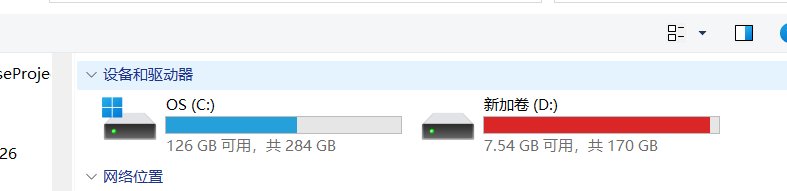






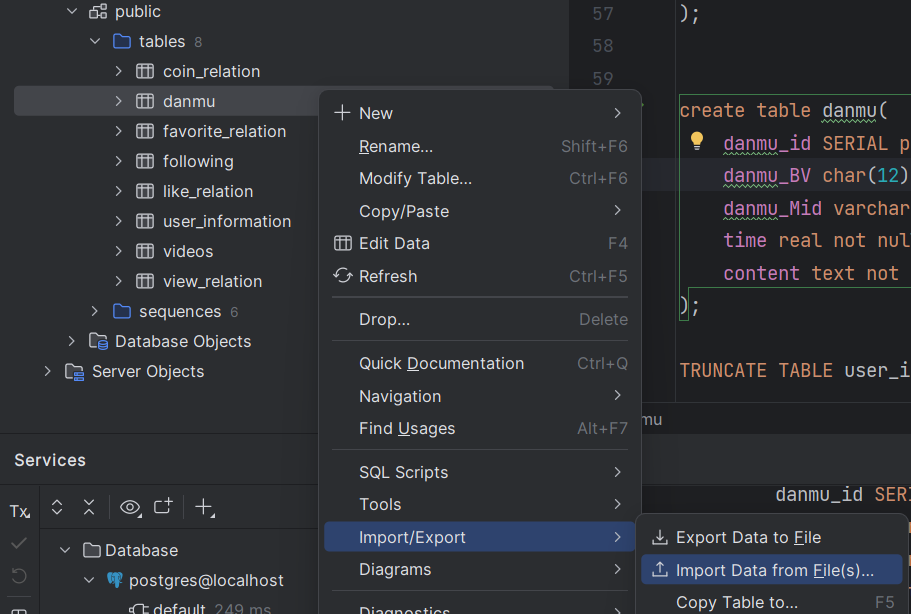




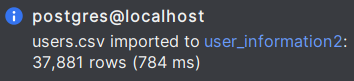


尝试利用不同方法导入csv文件

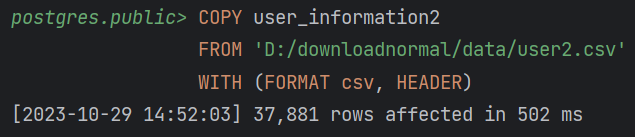
1：利用datagrip内置功能来导入数据。



导入user



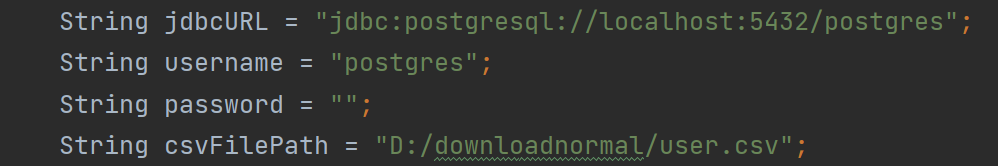
速度很快



用脚本



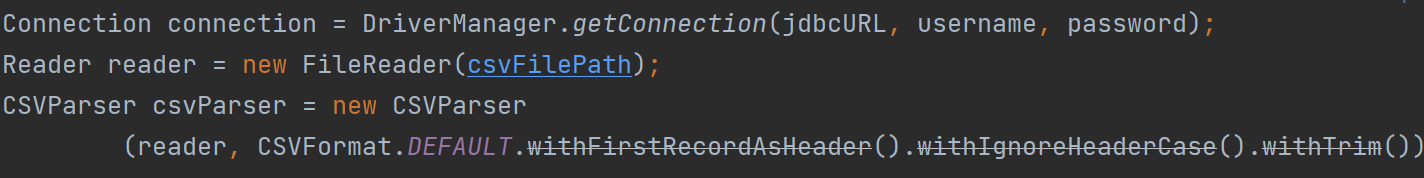
导入数据的过程:导入包（见最上面）



首先定义好JDBC（Java Database Connectivity）连接字符串，定义好数据库类型，ip地址和端口，数据库名称。

输入用户名和密码。（使用的数据库密码为空）

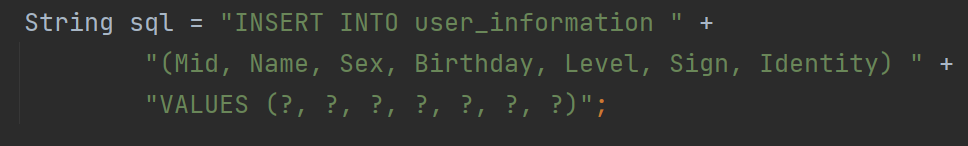
输入文件地址



建立数据库连接，并打开指向CSV文件的阅读器以及用于解析CSV内容的CSV解析器。



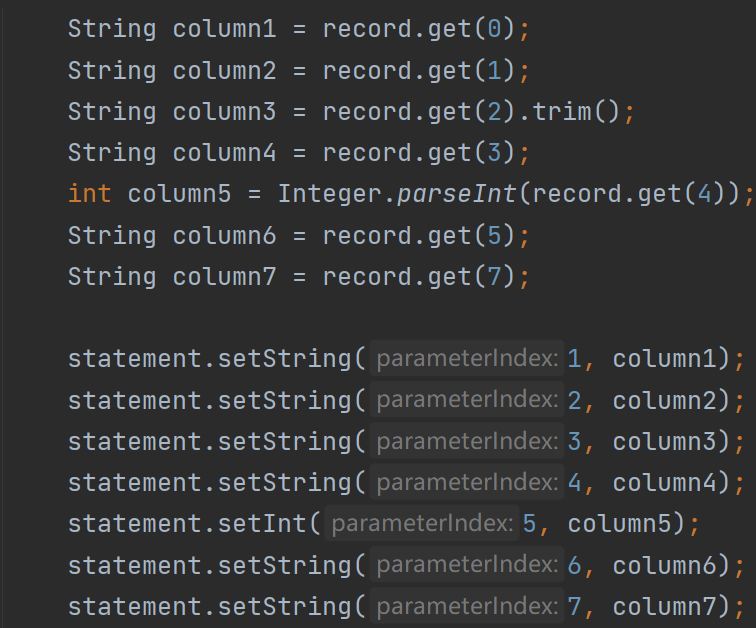
设置数据库事务的自动提交为假，以达到允许在后面进行批处理的效果。



准备SQL语句。



逐行遍历csv文件，对于每一行进行操作。

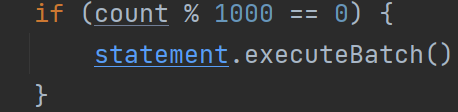


利用record.get(i)获取这一行中我们需要的信息。



在获取完信息后，利用addBatch将单个命令添加到批处理。

。



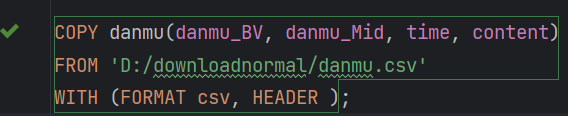
当批处理操作积累到1000个时我们一次性提交SQL命令到数据库。



最后提交事务，保存所有更改

利用copy指令来导入danmu





利用Java脚本

（3000次才做批处理）



1000次才做批处理）



（500次批处理）



（300次做批处理）



（100次做批处理）



不做批处理



利用buffreader +300次做批处理。



优化待续》。。。。

利用多线程：4个



6个



10个



20个



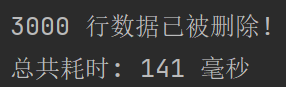
线程太高也没用

**CompletionService来在最后关闭线程池**



Delete\_group











删除弹幕，一行一行删。







Update：user



Insert,插入多少行

Insert 300：



Insert 3000



Insert30000



Insert300000



Select:

SELECT *count*(\*) from danmu where danmu\_mid='286952';

User的前300个用户发的弹幕综合

30



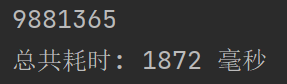
300



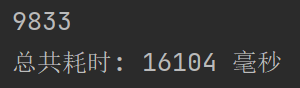
3000



30000



逐行select再相加：



速度慢很多

Select videos 的弹幕数量

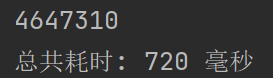
30

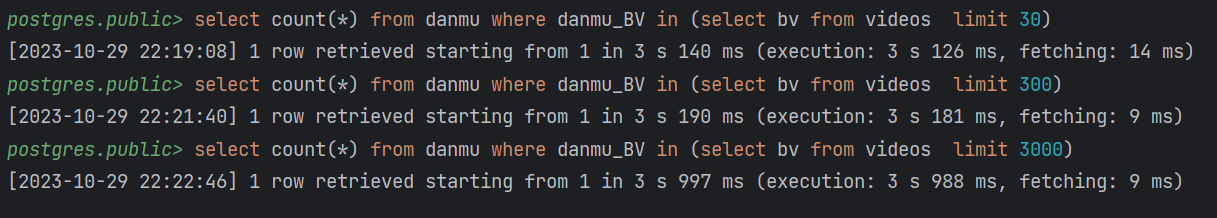


300



3000





用Java进行文件操作速度还比SQL语句快？

从数据库连接开始计时到数据库连接关闭：以下代码都是：

从打开文件流开始计时，到关闭文件流结束计时。

Insert

Insert:

一行一行插入

300000



这个确实太慢了，但是我比较了insert5行5行插入也没有快啊，没办法优化

30000



3000



300

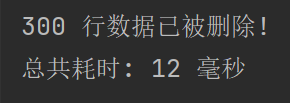


Delete：









Update：

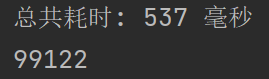
只能做到这个速度了

Select users

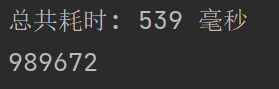
30



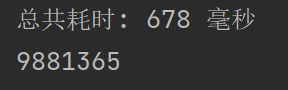
300



3000

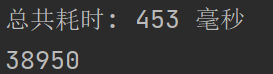


30000

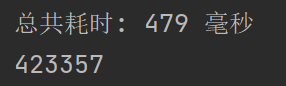


Videos

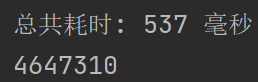
30



300



3000



配置：

设备名称 Fourwords

处理器 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz

机带 RAM 16.0 GB (15.7 GB 可用)

设备 ID 3943F5C8-9474-4A9A-B103-38492291162D

产品 ID 00342-30567-87083-AAOEM

系统类型 64 位操作系统, 基于 x64 的处理器

笔和触控 没有可用于此显示器的笔或触控输入

版本 Windows 11 家庭中文版

版本 22H2

安装日期 ‎2023/‎2/‎26

操作系统版本 22621.2428

体验 Windows Feature Experience Pack 1000.22674.1000.0

系统：Windows 11

固态硬盘（SSD）：1TB

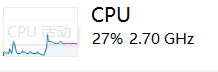
内存容量：32GB

处理高并发：



当开到100个线程时，电脑性能已经不够用了

在设计数据库架构时可以用队列批量处理，比如使用ArrayBlockingQueue可以实现队列的批量操作，逻辑：每次数据来了放入队列，启动一个消费线程，从队列中一次性拉取所有，使用batch操作，同样可以减少对数据库的并发。对cpu占用减少，但是速度变慢



10000次



使用存储过程，将一次请求多次访问数据库操作的业务放在存储过程中，这样可以一次访问多次操作

将逻辑放在数据库服务器上（例如，使用存储过程）确实可以在某些情况下帮助处理高并发，但它不是万能的。这种方法对高并发有帮助的原因主要是减少了网络往返和应用层的处理时间，从而可以更快地响应单个请求。这意味着服务器可以在同一时间内处理更多的请求。特别是在以下情况下，这种方法可能特别有用：

网络延迟显著：当应用程序和数据库服务器在不同的物理位置，且网络延迟显著时，减少往返次数可以大幅度提高性能。

大量的小型查询：如果应用程序需要发送大量的小型查询，合并这些查询到单个的数据库调用可以减少网络和应用层的开销。

复杂的数据处理逻辑：当需要进行复杂的数据处理，且这些处理主要依赖数据库中的数据时，数据库端的处理可能会更加高效。

反作用：

增加数据库负载：所有的计算和逻辑处理都在数据库端进行，这可能会增加数据库的CPU和内存负载，成为性能瓶颈。

减少应用层的可伸缩性：应用服务器通常更容易水平扩展（增加更多的服务器实例）来处理更多的并发请求，而数据库通常更难以这样做。

速度没差别，可能是因为应用和数据库在一个服务器上

尝试select多次:  
10次：

添加索引后

10次



10000次：



一百万次：



要在合适的地方添加索引，比如说danmu，user，videos

但我要查寻哪个表就必须在那个表上添加索引，与外键无关

索引不是越多越好，影响插入更新效率

Mysql （6个线程时）多线程表现不是很好，极大占用电脑资源：



一般导入



有点慢

Select

Mysql 不加索引10次



加索引10



10000



**100万次**



**慢很多**