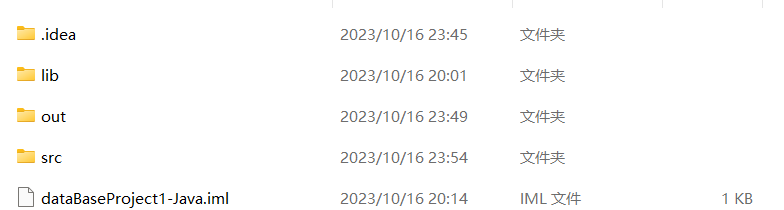
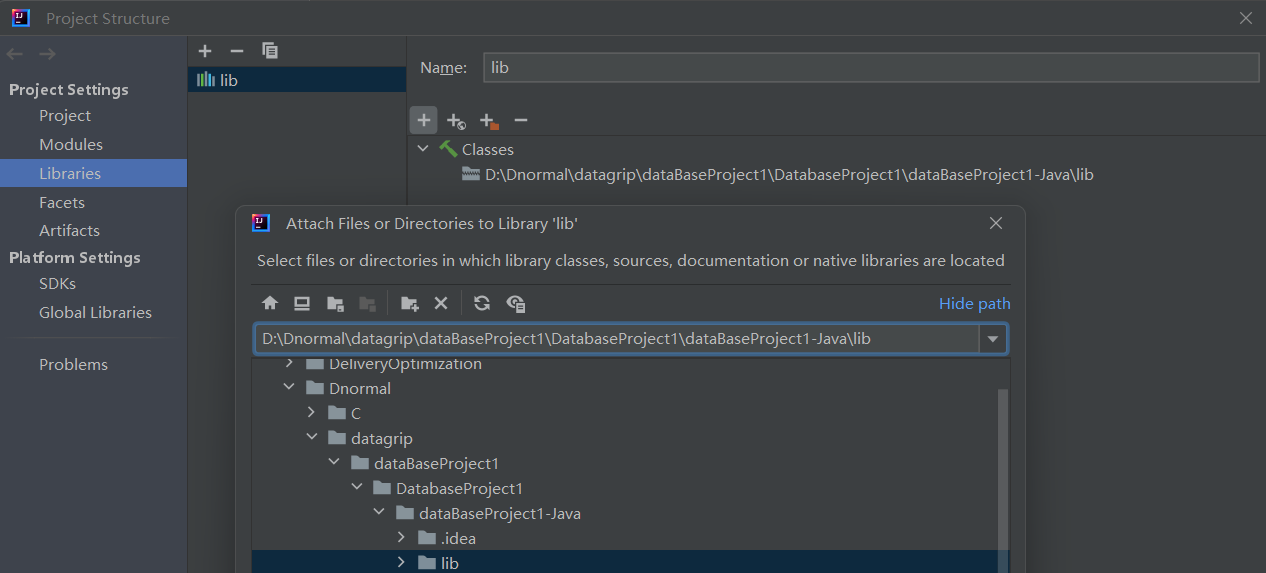
下载连接postgres对应的jar包

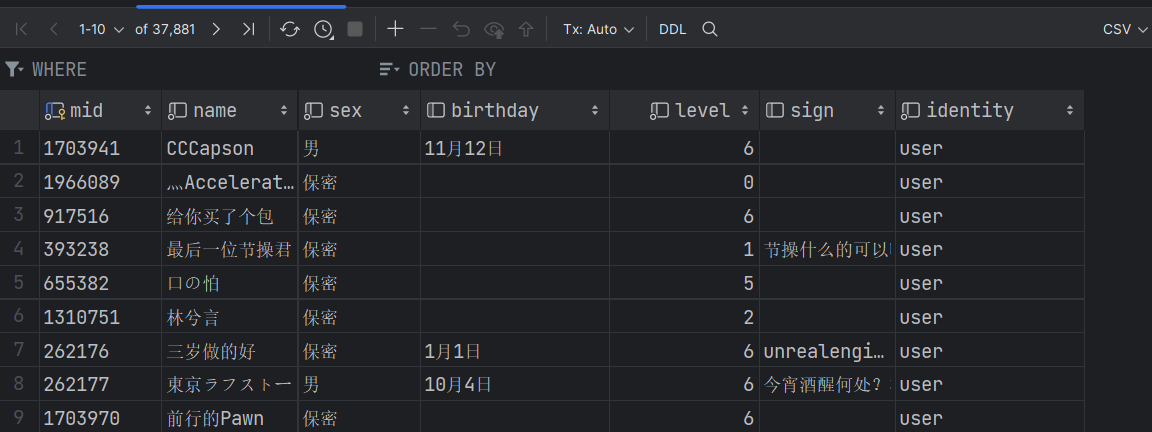


在Java项目目录下新建一个lib文件夹，并将jar包放入



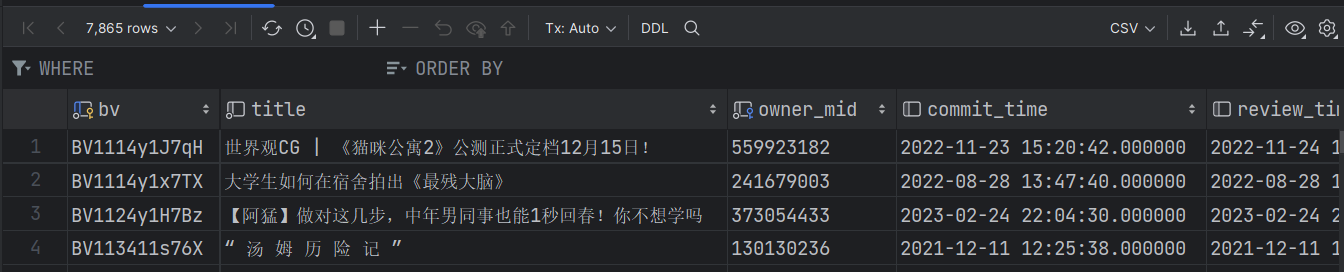
将lib文件夹添加到项目的libraries中

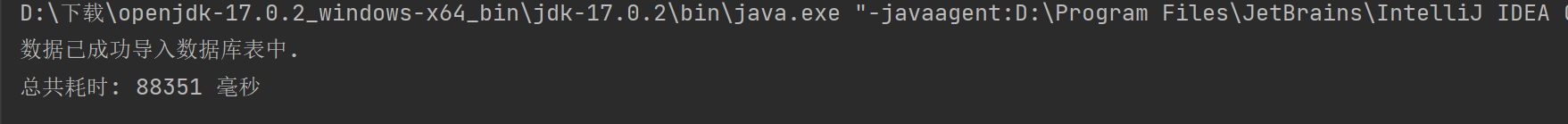


导入user后结果：  


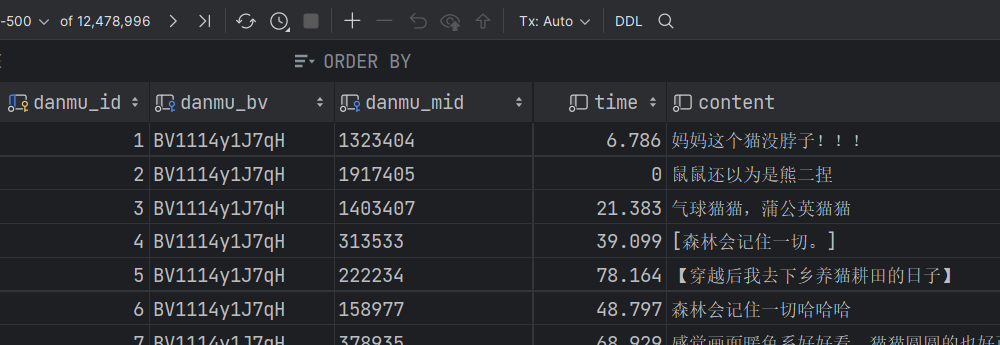


导入video后结果：



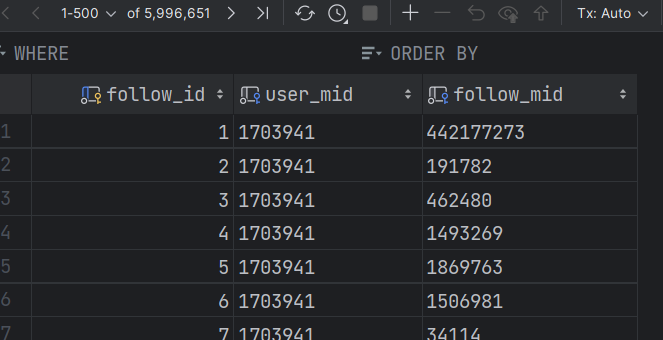


弹幕



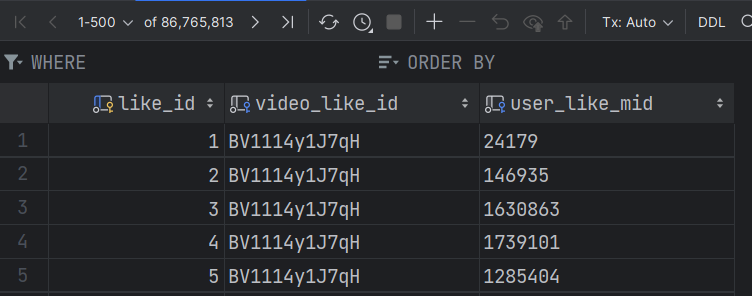


Following



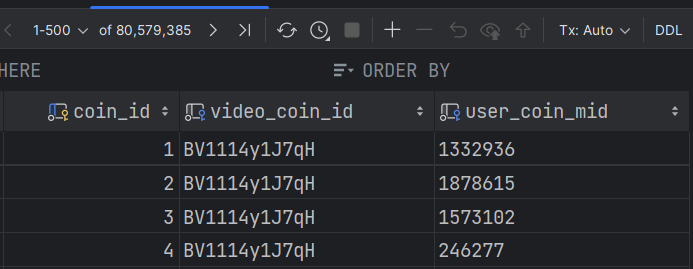


Like relation

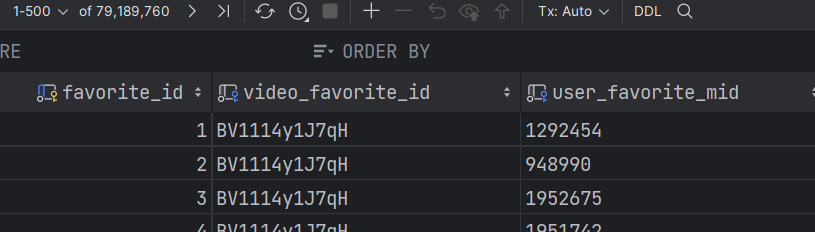


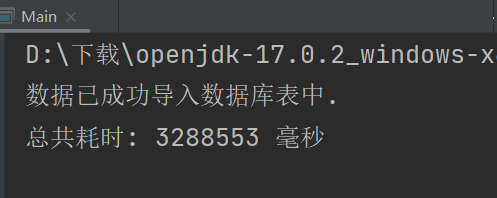


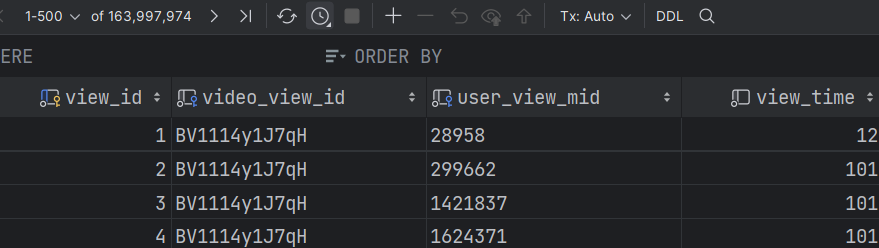
最大堆内存开到10g了



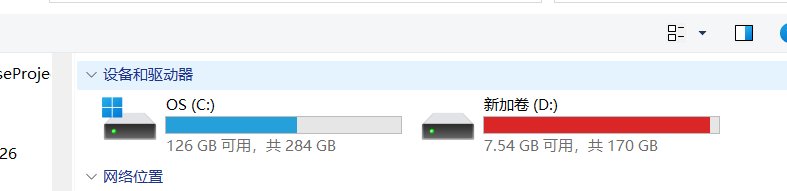






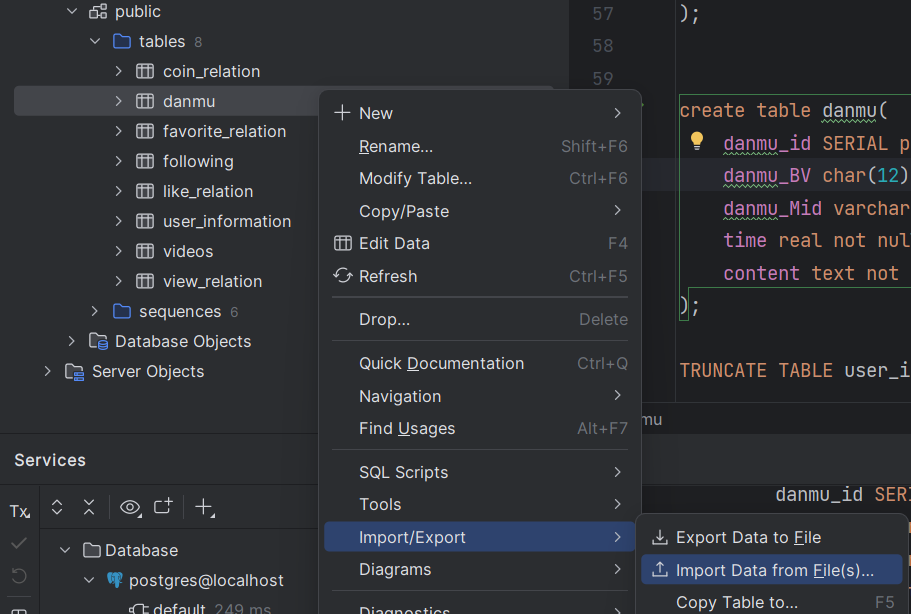




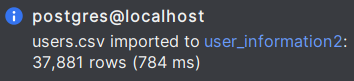


尝试利用不同方法导入csv文件

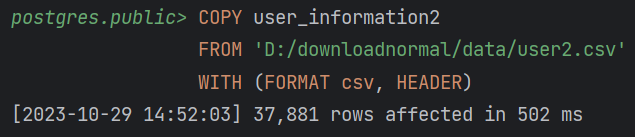
1：利用datagrip内置功能来导入数据。



导入user



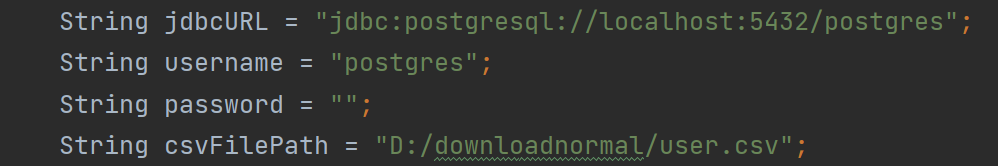
速度很快



用脚本



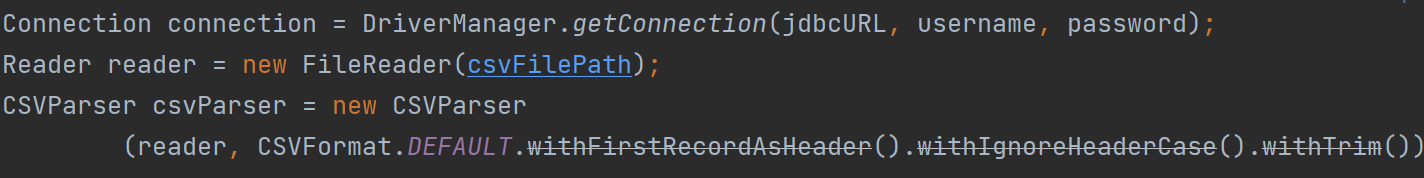
导入数据的过程:导入包（见最上面）



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）

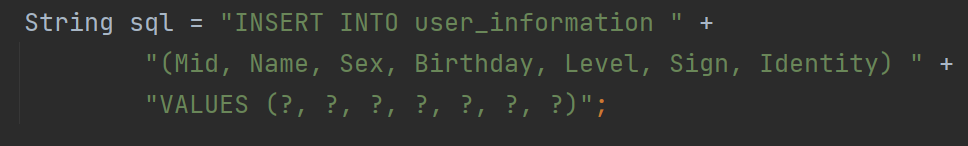
输入文件地址



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。



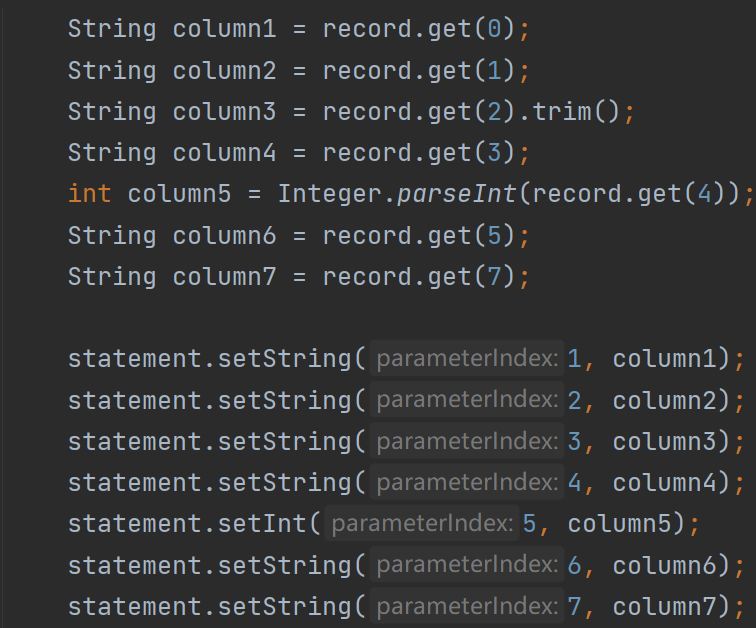
设置数据库事务的自动提交为假，以达到允许在后面进行批处理的效果。



准备SQL语句。



逐行遍历csv文件，对于每一行进行操作。

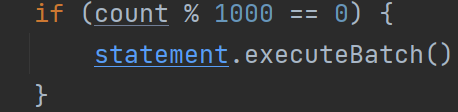


利用record.get(i)获取这一行中我们需要的信息。



在获取完信息后，利用addBatch将单个命令添加到批处理。

。



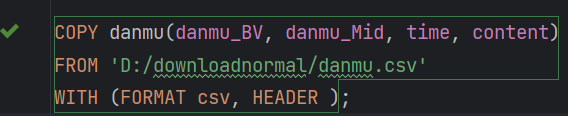
当批处理操作积累到1000个时我们一次性提交SQL命令到数据库。



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



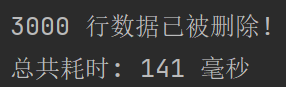
线程太高也没用

**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



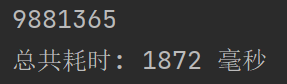
300



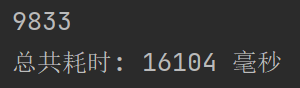
3000



30000



逐行select再相加：



速度慢很多

Select videos 的弹幕数量

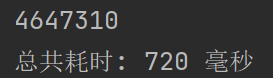
30

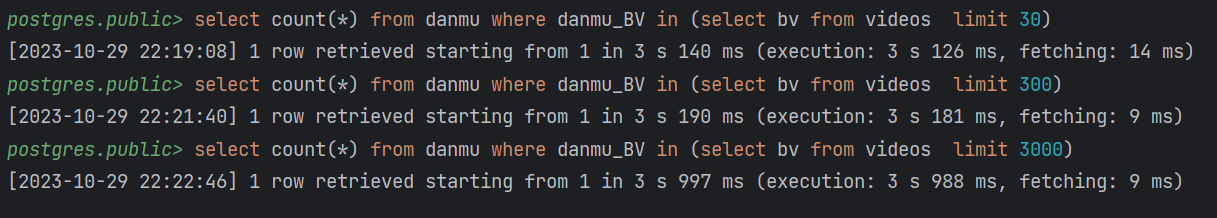


300



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

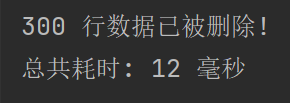


Delete：









Update：

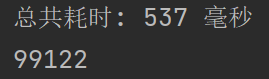
只能做到这个速度了

Select users

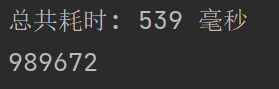
30



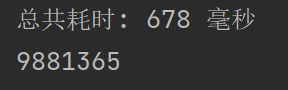
300



3000

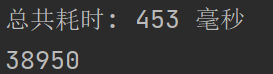


30000

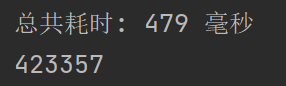


Videos

30



300



3000

