待完成：

```
1.需要完善任务3的文档 对应文档、数据、加入压缩包后文档中也需要描述

2.任务3-2的不同语言部分
```

项目名称：2023年春季CS307课程项目（第一部分）

主要贡献者、分工：

12110416 刘家宝：数据库初步设计；绘制ER图；建表初步语句；数据库性能测试；测试数据生成

12112712 莫羡瑜：完善数据库设计；完善建表语句；导入数据；数据库性能测试；改善导入数据算法

实验课时：周四三四节　贡献比：50%/50%

计算机配置：操作系统：Windows10 CPU: i7-11800H 使用语言：java 17; python 3.9.12

任务1：绘制E-R图

根据ER图模拟构建思路

1.1 建立实体

1.作者（Author）2.文章（Post）3.评论（Reply）4.次级评论（Secondary Reply）5.分类（Category）6.城市（Posting City）

1.2 讨论关系

一对多关系：

1. 作者（Author）与文章（Post）：一个作者可以发布多篇文章，而每篇文章只能有一个作者。
2. 文章（Post）与评论（Reply）：一篇文章可以有多个评论，而每个评论只能针对一篇文章。
3. 评论（Reply）与次级评论（Secondary Reply）：一个评论可以有多个次级评论，而每个次级评论只能针对一个评论。
4. 城市（Posting City）与文章（Post）：一个城市可以有多篇文章发布，而每篇文章只能发布在一个城市。

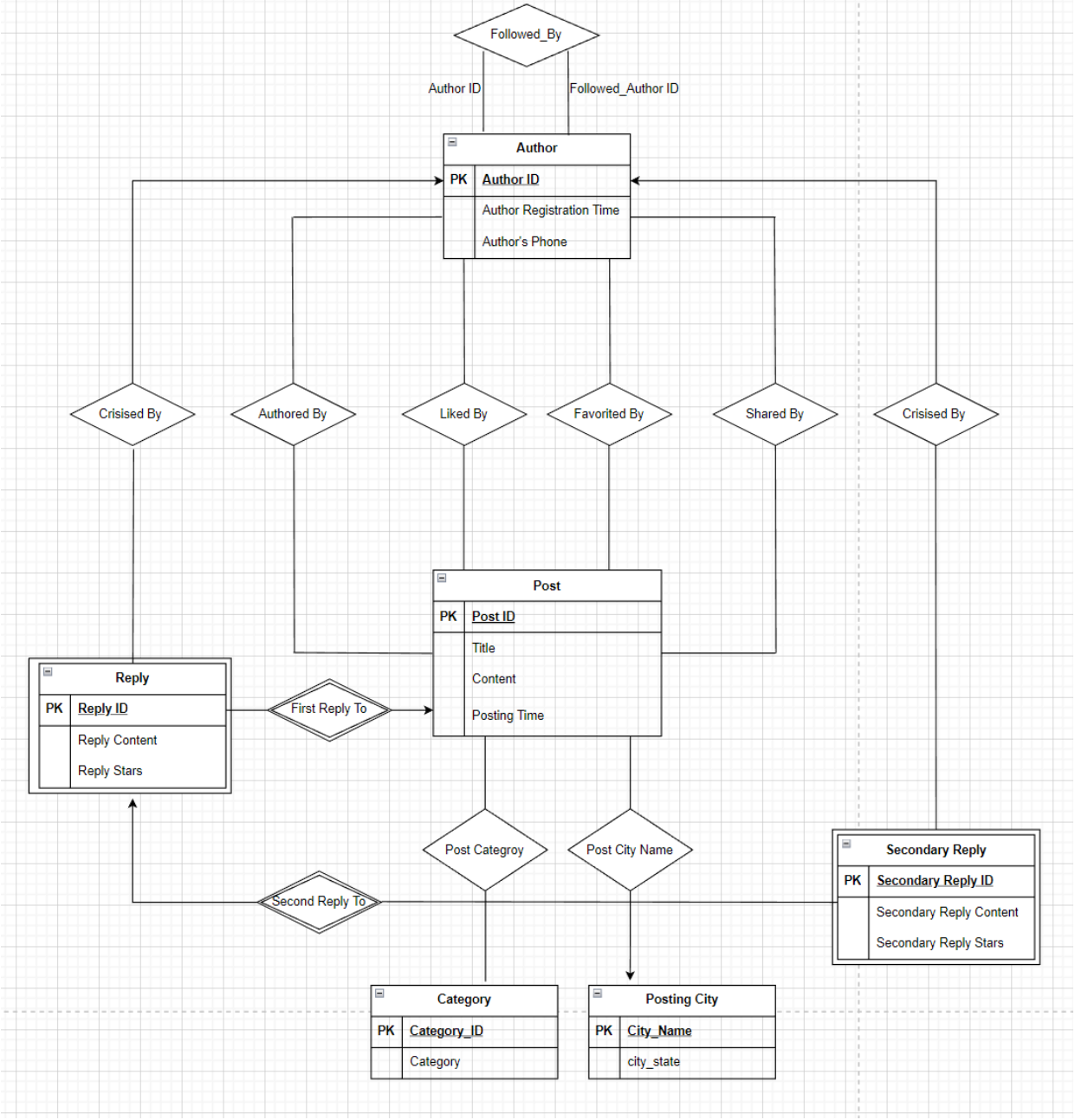
多对多关系：

1. 作者（Author）与关注的作者（Followed Author）：一个作者可以关注多个作者，同时也可以被多个作者关注。
2. 作者（Author）与收藏的文章（Favorite Post）：一个作者可以收藏多篇文章，同时一篇文章也可以被多个作者收藏。
3. 作者（Author）与分享的文章（Shared Post）：一个作者可以分享多篇文章，同时一篇文章也可以被多个作者分享。

4. 作者（Author）与点赞的文章（Liked Post）：一个作者可以点赞多篇文章，同时一篇文章也可以被多个作者点赞。
5. 文章（Post）与分类（Category）：一篇文章可以属于多个分类，同时一个分类也可以包含多篇文章。

1.3 ER图：

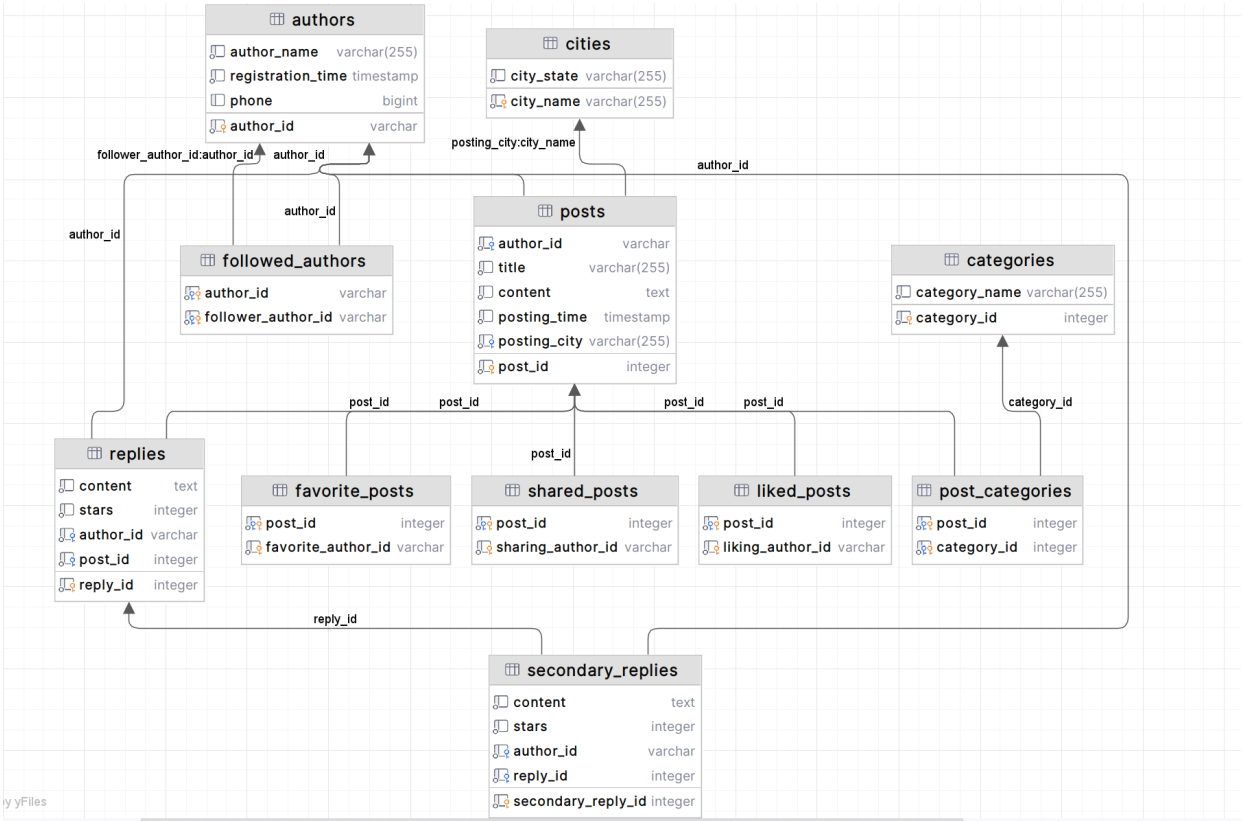
使用 drawio 进行ER图绘制



任务2：关系数据库设计

Data Grip生成E-R图快照

根据任务一中分析，写出相应sql语句，由Data Grip生成的E-R图的快照如下：



相关建表语句请见：

2.1、每个表格和列的含义

主体表：

1. 作者表（authors）

   author_id：作者的ID，主键　　author_name：作者的名字，不能为空且唯一

   registration_time：作者的注册时间，不能为空　phone：作者的电话号码，唯一

2. 文章表（posts）

   post_id：文章的唯一标识符，主键　　author_id：与作者表关联的外键，表示文章作者的ID

   title：文章的标题，不能为空　　content：文章的内容，不能为空

   posting_time：文章发布的时间，不能为空　　posting_city：文章发布的城市的名字，与城市表关联的外键

3. 评论表（replies）

   reply_id：评论的唯一标识符，主键　　content：评论的内容，不能为空

   stars：评论的点赞数，不能为空　　author_id：与作者表关联的外键，表示评论作者的ID

   post_id：与文章表关联的外键，表示被评论的文章的 ID。

4. 次级评论表（secondary_replies）

   secondary_reply_id：次级评论的唯一标识符，主键　content：次级评论的内容，不能为空

stars：次级评论的点赞数，不能为空　　author_id：与作者表关联的外键，表示次级评论作者的 ID。

reply_id：与评论表关联的外键，表示被次级评论的评论的 ID。

5. 分类表（categories）

   category_id：分类的唯一标识符，主键　　category_name：分类的名称，不能为空且唯一

6. 城市表（cities）

   city_name：城市的名称，主键　　city_state：城市所在的国家，不能为空

关系表：

1. 关注表（followed_authors）

   author_id：与作者表关联的外键，表示author的 ID

   follower_author_id：表示author关注的author的 ID，同时与作者表关联的外键，不能为空

2. 收藏表（favorite_posts）

   post_id：与文章表关联的外键，表示被收藏的文章的 ID

   favorite_author_id：收藏者的作者 ID，不能为空

3. 分享表（shared_posts）

   post_id：与文章表关联的外键，表示被分享的文章的 ID

   sharing_author_id：分享者的作者 ID，不能为空

4. 点赞表（liked_posts）

   post_id：与文章表关联的外键，表示被点赞的文章的 ID

   liking_author_id：点赞者的作者 ID，不能为空

5. 文章分类表（post_categories）

   post_id：与文章表关联的外键，表示文章的 ID，不能为空

   category_id：与分类表关联的外键，表示分类的 ID，不能为空

关于部分扩展性的额外解释：

（1）考虑到作者想改名的需求，利用authorID作为关联post与author，以及关系表中对应的是作者ID，当作者想改名的时候只需更改author表中对应ID的名字即可。

（2）将category与city从post分离出来，既满足了原子性，又描述了post相关属性，还可拓展成新情况。

（3）将replies.json中的一级回复和二级回复分割出来，减少了信息的冗余度，方便修改相关信息。

任务3：数据导入

在此任务中，我们使用java和python两种编程语言，对数据进行导入，并分析相关导入性能，以及优化算法。

相关文件请见：

3.1、基本数据导入

1. 在未对数据进行预处理的情况下使用java中JDBC的prepare策略。
2. 在java中人工设置Map authorNameAndID,Map cities和List categories方便处理相关table。
3. 导入数据顺序按照Data Grip生成的ER图中表与表之间的层级关系。

3.2、算法优化：

（1）我们通过对比不同编程语言同种策略，同种语言不同策略来对比分析。

（2）由于在实验过程中发现，在没有对数据进行预处理(NoPretreatment包下的代码)的情况下，用脚本边导入边判断，判断所耗时间指数增加，会使得导入效率随数据的增加而指数级降低，致使在单线程的情况下，不同策略的相对效率差异逐渐减小，在选择事务策略的情况下，单线程与多线程的效率差异逐渐变为0。原因是因为作者信息存在过多的冗余（即该作者并没有发过post，只对post进行点赞，评论等）。因此，我们使用Pretreatment包下的代码进行算法优化分析，以下实验的时间均是在预处理后直接的导入时间。

（3）每次实验迭代三次，取三次实验时间平均值。

（4）对于测试数据，我们编写了两个python脚本进行数据的扩展，生成原始数据5倍、10倍的数据。

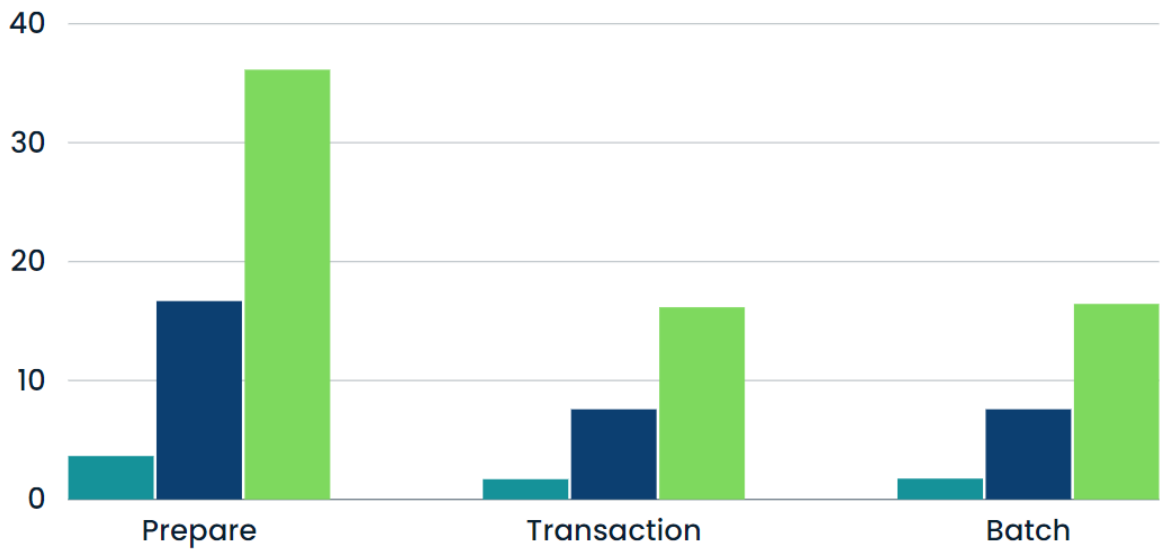
（5）注：运行结果控制台输出放在压缩包，下面文档中只放结果

1.同种语言不同策略

（1）单线程（3.4w 条数据，15.8w 条数据，33.9w 条数据）

a. Prepare实现　b. Transaction实现　c. Batch实现

	3.4w	**15.8w**	**33.9w**
Prepare	3.645 s	16.652 s	36.078s
Transaction	1.693s	7.552 s	16.101 s
Batch	1.743 s	7.553 s	16.381 s

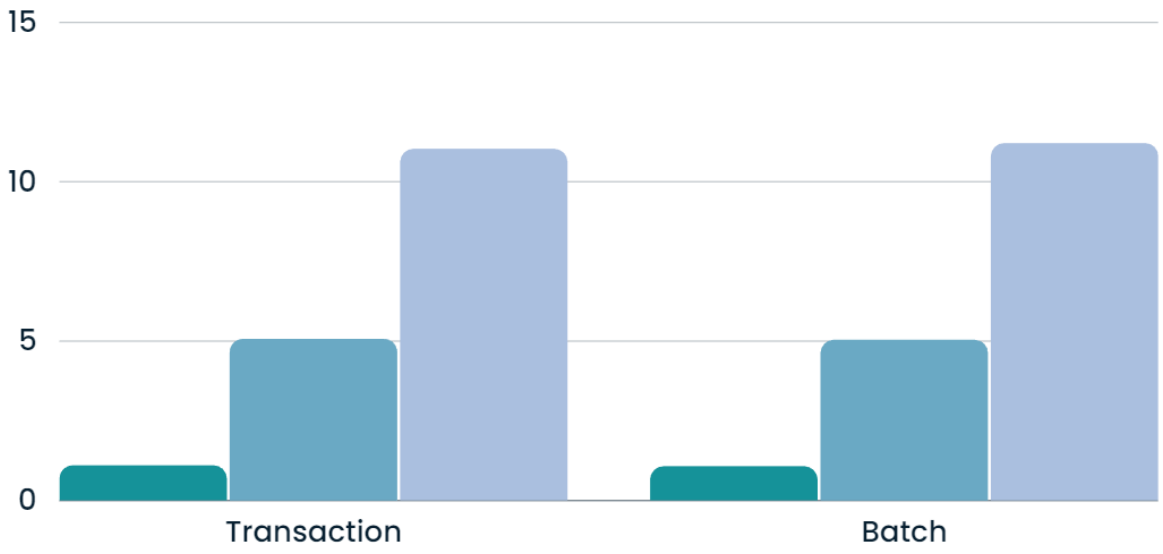


实验结论：在当前测试环境下，我们发现对于单线程而言，Transaction策略的效率会比Prepare高，Batch策略和Transaction相差不大。同时，单线程中，各个策略效率大致上随数据的增加线性增加。

（2）多线程(对比transaction和batch，3.4w 条数据，15.8w 条数据，33.9w 条数据)

a. Transaction + 多线程 b. Batch + 多线程

	3.4w	**15.8w**	**33.9w**
Transaction	1.104 s	5.062 s	11.013 s
Batch	1.074 s	5.034 s	11.194 s



实验结论：由实验数据可得在原始数据、5倍原始数据、10倍原始数据下，相比于单线程而言，多线程的效率大大提高。随着数据量的增加，多线程能带来的提升明显增加。同时，多线程中，各个策略效率大致上随数据的增加**线性增加**。此外，这里的多线程只是分级同时导入所有表的数据，对于单张表，里面还可细分成多个线程同时导入数据。在多线程导入数据的时候打开任务管理器，发现实际上CPU的占用平均在40%左右，最高达88%，推测在单张表里细分多个线程导入数据可提高导入效率。

2.不同语言同种策略（待完善）

Python使用psycopg2包，其底层调用的是C语言库；Java使用JDBC，底层调用的是java语言。我们采用transaction策略。

（1）Java

（2）Python

任务总结：