数据库原理 Project1 报告

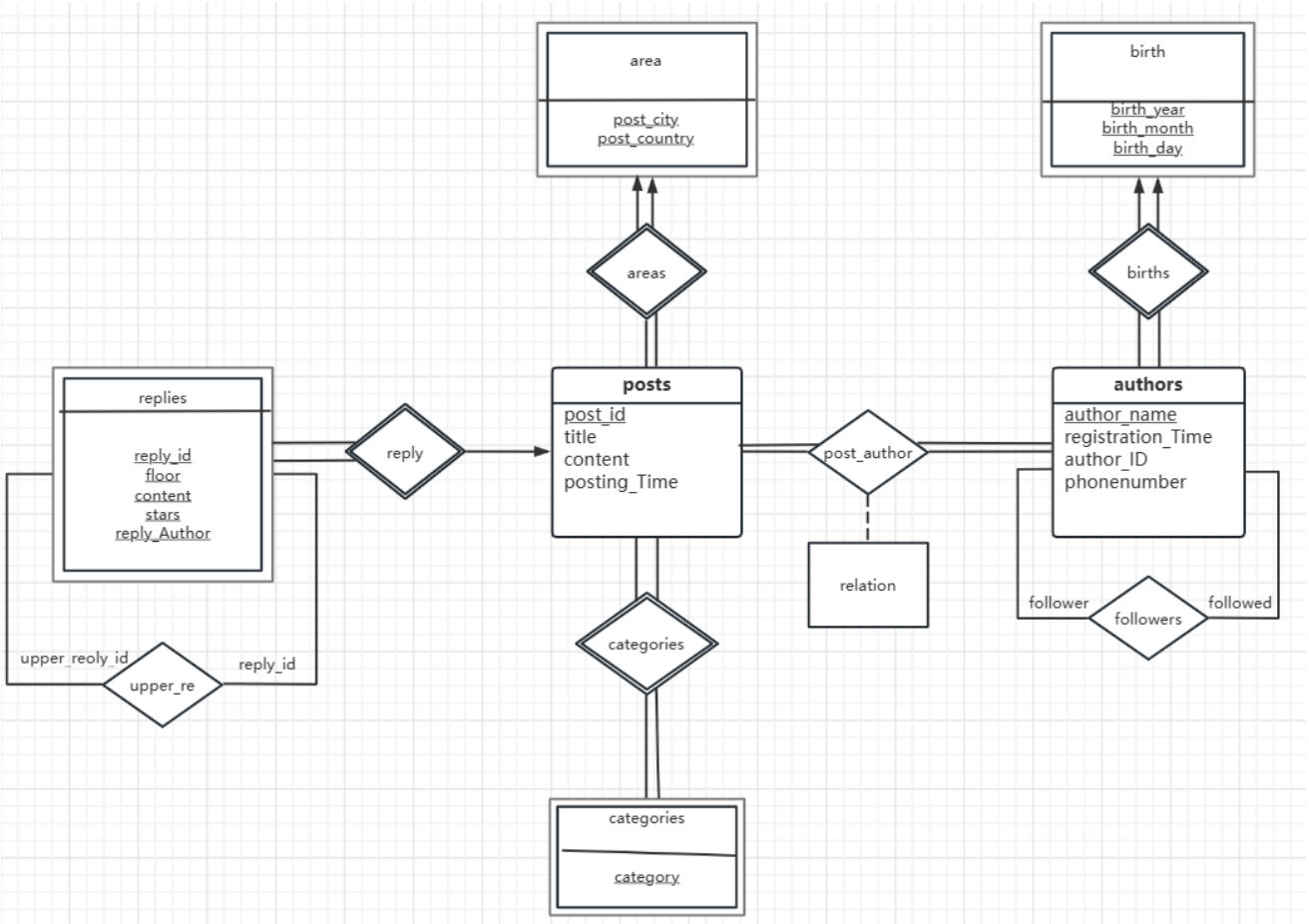
第一部分：项目概述

数据库原理 project1 基于 posts.json 和 replies.json 两个文件读取出 post、replies、author 等相关数据信息，通过建立 E-R 图确定各数据间的相关关系并设计创造各关联表，使用 Java 语言多种方式导入数据实现优化，最终完成数据库设计。

该项目是由贺小珊(学号:12110848)和匡晟(学号:12011130)共同完成的(Group Number：217），贺小珊同学负责 E-R 图的设计与绘制，匡晟同学负责数据库表的设计与创建,两人共同完成数据的导入与测试比较,贡献比分别为 50%和 50%。

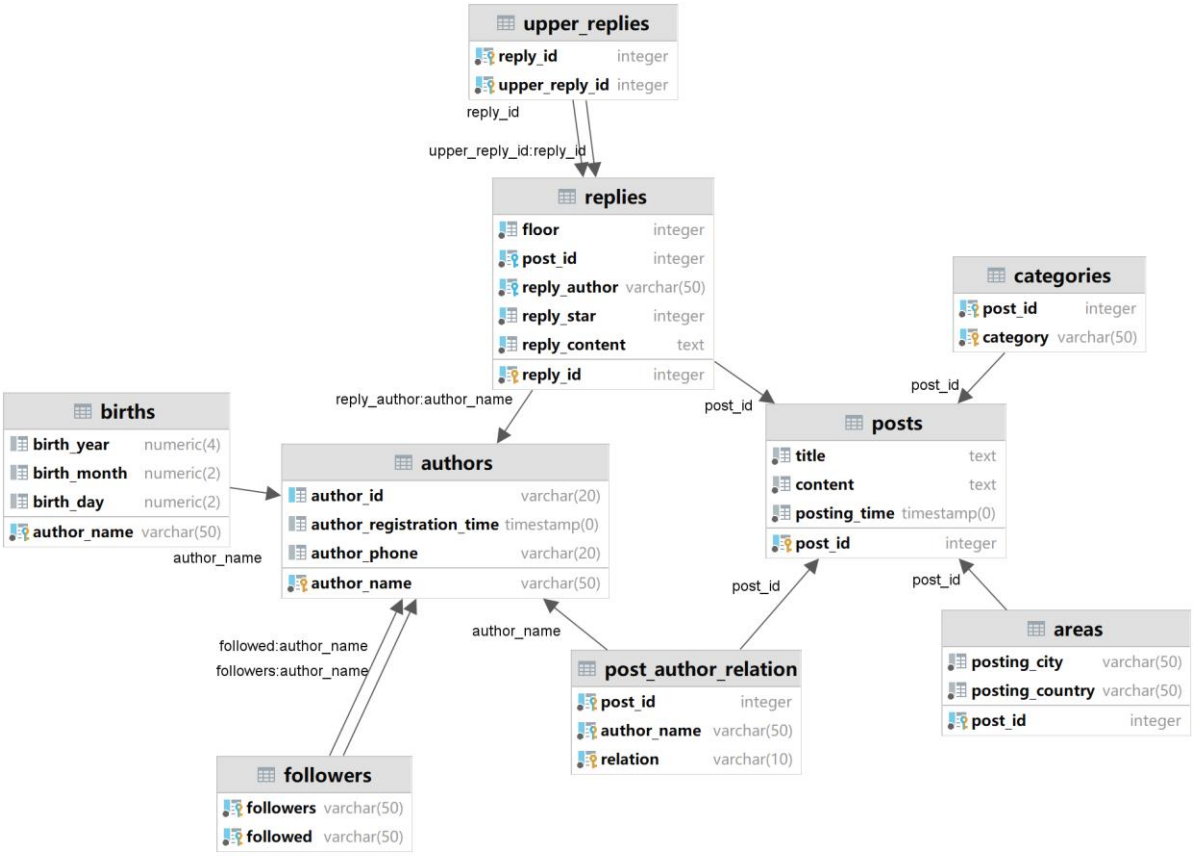
第二部分：项目实现

任务一：E-R 图绘制 （使用的作图软件为 ProcessOn）



任务二：关联数据库设计

1. DataGrip 生成的 E-R 图



2. 表设计（建表文件见附件.sql 文件）

authors 表

author_name：作者名字
author_id：作者 ID
author_rigistration_time：作者注册时间（为 timestamp 数据类型，精确度到秒）
author_phone：作者电话
Primary key：author_name
Unique：author_id
Not null：author_name
说明：此表中前 203 条数据均来自于 posts.json 文件，此后的数据唯有 author_name 来源两个文件，其余均为随机生成，且保证每一个作者的注册时间早于其出现的所有帖子的发布时间，且保证所有作者的 ID 中隐含的出生日期早于该账号注册时间十年（即最小十岁发帖）。此外，不保证随机生成的电话号码不重复，即意味着同一个电话号码可以注册多个账号。

births 表

author_name：作者名字（外键，指向 authors 表中 author_name）
birth_year：作者出生年份（为 numeric 数据类型，最多 4 位）
birth_month：作者出生月份（为 numeric 数据类型，最多 2 位）
birth_day：作者出生日期（为 numeric 数据类型，最多 2 位）
Primary key：author_name
Not null：author_name
说明：作者的出生日期由 authors 表中 author_id 推出

followers 表

followers：关注者的名字（外键，指向 authors 表中 author_name）
followed：被关注者的名字（外键，指向 authors 表中 author_name）
Primary key：（followers, followed）
Not null：followers, followed
说明：此表中所有数据由 posts.json 文件中的 Author_Followed_By 得出

posts 表

post_id：帖子 ID（为 int 数据类型）
title：帖子标题（为 text 数据类型）
content：帖子内容（为 text 数据类型）
posting_time：帖子发布时间（为 timestamp 数据类型，精确度到秒）
Primary key：post_id
Not null：post_id, title, content, posting_time
说明：此表中所有数据由 posts.json 文件得出

areas 表

post_id：帖子 ID（外键，指向 posts 表中 post_id，为 int 数据类型）
posting_city：帖子发布城市
posting_country：帖子发布国家
Not null：post_id, posting_city, posting_country
说明：此表中所有数据由 posts.json 文件得出，原文件中 Posting_City 包括城市与国家，为了满足第一范式，将之分为两列

categories 表

post_id：帖子 ID（外键，指向 posts 表中 post_id，为 int 数据类型）
category：帖子种类
Primary key：（post_id, category）
Not null：post_id, category
说明：此表中所有数据由 posts.json 文件得出，原文件中 Category 包含多个，为了满足第一范式，经过分析，将多对多关系表示为二元表

post_author_relation 表

post_id：帖子 ID（外键，指向 posts 表中 post_id，为 int 数据类型）
author_name：作者名字（外键，指向 authors 表中 author_name）
relation：帖子与作者的关系

Primary key：（post_id, author_name, relation）
Not null：post_id, author_name, relation
说明：此表中所有数据由 posts.json 文件得出，若 relation 为‘P’，作者 post 该贴（即为该贴作者）；若 relation 为‘F’，作者 favorite 该贴；若 relation 为‘S’，作者 share 该贴；若 relation 为‘L’，作者 like 该贴

replies 表

reply_id：回复 ID（为 int 数据类型）
floor：回复楼层（为 int 数据类型）
post_id：回复所属帖子 ID（外键，指向 posts 表中 post_id，为 int 数据类型）
reply_author：回复作者（外键，指向 authors 表中 author_name）
reply_star：回复星数（为 int 数据类型）
reply_content：回复内容（为 text 数据类型）
Primary key：reply_id
Unique：（post_id, floor, reply_author, reply_star, reply_content）
Not null：reply_id, floor, post_id, reply_author, reply_star, reply_content
说明：该表中中文“回复”均为名词，指该条回复，非动词，下表同。该表中 reply_id, floor 为经过处理得到，每一条回复都有一个独一无二的 reply_id, floor 代表所属楼层，本次 project 中最大为 2，且 1 楼回复 1229 条，2 楼回复 3009 条。
其余数据由 replies.json 文件得出。在 unique 约束中，本组将 reply_content 包含进来，虽然插入将会变慢，但是本组认为数据的精确性更为重要

upper_replies 表

reply_id：回复 ID（外键，指向 replies 表中 reply_id，为 int 数据类型）
upper_reply_id：回复的上楼回复 ID（外键，指向 replies 表中 reply_id，为 int 数据类型）
Primary key：（reply_id, upper_reply_id）
Not null：reply_id, upper_reply_id
说明：该表中所有数据均由 replies.json 文件经过处理得出

3. 额外说明

数据来源：所有表中数据均直接或间接来源 posts.json 和 replies.json 两个文件

三大范式：本次项目数据库设计满足三大范式，如 areas 表中 posting_city 与 posting_country 分开满足第一范式，使列具有原子性；每张表都有一个主键，非主键类必须完全依赖于主键，而不能只依赖主键的一部分，满足第二范式；如 authors 表中有 author_id 列，可以推出 author 的出生年月日，非主键列必须直接依赖于主键，不能存在传递依赖，故制成 births 表，满足第三范式

外键指向：每张表要有外键，或者有其他表的外键指向，且未成环

特殊约束：每一张表都至少有一列非空，且除了主键自增的 id 之外，有其他 unique 约束的列，若无特殊说明 unique 列，主键即为 unique 列，需要说明的是，打括号意为联合主键或联合约束，未打括号意为单独主键或单独约束

数据类型：表设计的所有列，除已说明数据类型的列外，其余均为 varchar 数据类型

可拓展性：本次数据库设计可拓展性较强，如在 replies 表中设 floor 列，若此后有三级回复、四级回复等可直接插入 replies 表中，并在 upper_replies 中插入该回复及其上层回复即可，无需单独建表

用户属性：本次数据库设计是面向使用对象的，所以单独列出 births 表和 areas 表，方便用户分析发帖作者年龄结构、地域属性等

任务三：数据导入

1. 基础导入

脚本文件

见附件 LoaderAuthors.java、LoaderFollowers.java、LoaderBirths.java、LoaderPosts.java、LoaderCategories.java、LoaderAreas.java、LoaderPostAuthorRelatiob.java、LoaderReplies.java、LoaderUpperReply.java、AuthorGenerator.java、IDGenerator.java、Post.java、Replies.java、SingleReply.java 文件

导入描述

本组将 Java 语言作为客户端导入数据，使用的是课上的第四种方式 Transaction，即导入时只创建一次数据库链接，一次性编译 sql 语句，每一次执行时直接用编译好的内容传参，采用事物处理，每执行一条 sql 语句，在缓冲区里执行内容，当再次执行 commit 时一次性写入硬盘，提高访问硬盘效率，单线程操作，没有进行批处理。
本次数据库共创建九张表，故分九次导入数据。
a. 在 DataGrip 中运行附件中的 Project1.sql 文件完成建表，由于 Java 文件中无建表、清表操作，故若需重新导入数据，需先删表再建表
b. 在 IDE 运行 AuthorGenerator.java 文件，给未在 posts.json 的 Author 部分的作者随机生成一个 ID 和注册时间，满足条件已在上一部分详述，并将所有 Author 写入 Authors.txt 文件，最后运行 LoaderAuthors.java 文件，需要说明的是，由于事先已生成 Authors，故无需再运行 AuthorGenerator.java 文件

c. 依次运行 LoaderFollowers.java 、 LoaderBirths.java 、 LoaderPosts.java 、 LoaderCategories.java、LoaderAreas.java、LoaderPostAuthorRelatiob.java、LoaderReplies.java、LoaderUpperReply.java 文件，因为各表之间存在相关联的外键指向，故导入顺序很重要且不能变

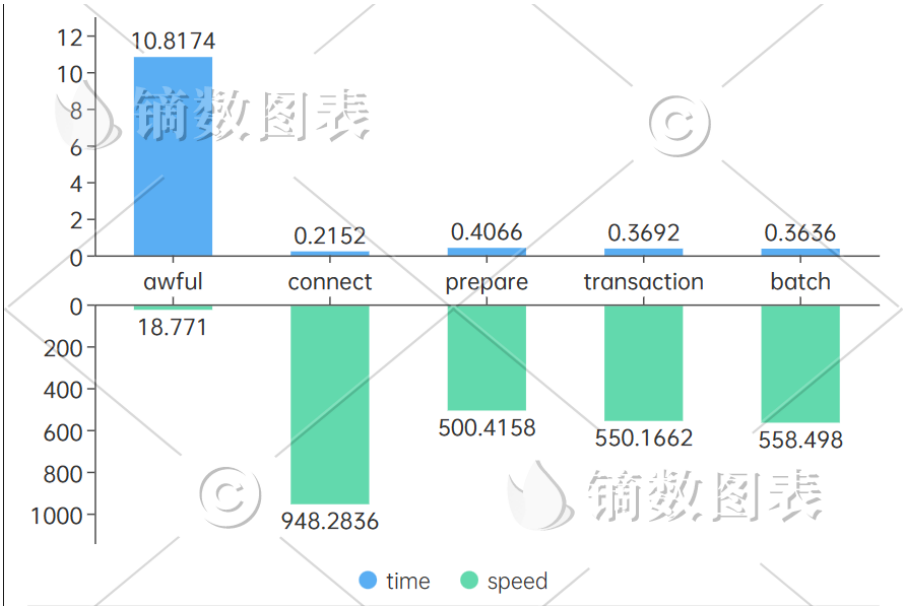
2. 优化比较

测试说明

我们采用了五种导入方式，以下用字母代号表示。
A：Awful，每一次导入创建一次数据库链接
C：Connect，导入时只创建一次数据库链接
P：Prepare，在 C 的基础上一次性编译 sql 语句
T：Transaction，在 P 的基础上提高访问硬盘效率
B：Batch，在 T 的基础上进行批处理
同时，测试的时间单位为秒，速度单位为条每秒，测试数据见附件.xlsx 文件

初级测试

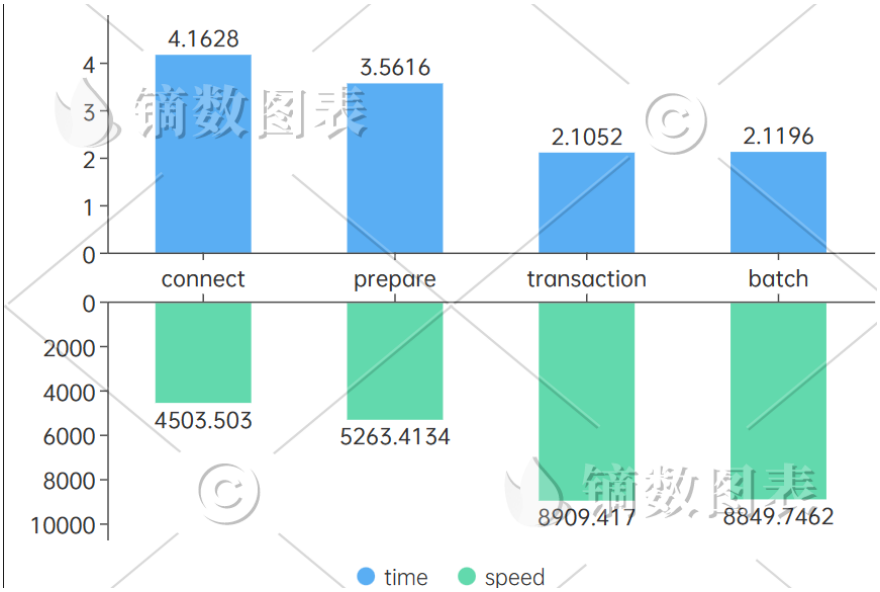
第一次测试我们导入数据量最小的 areas 表，数据量为 203 条，每一种方法都测试五次（测试文件即附件中的 test1Awful.java、test2Connect.java、test3Prepare.java、test4Transaction.java、test5Batch.java 文件），取平均值，统计图如下



显然，A 方法速度过慢，我们将其淘汰，C 方法最快，但数据量过小，误差可能较大

中级测试

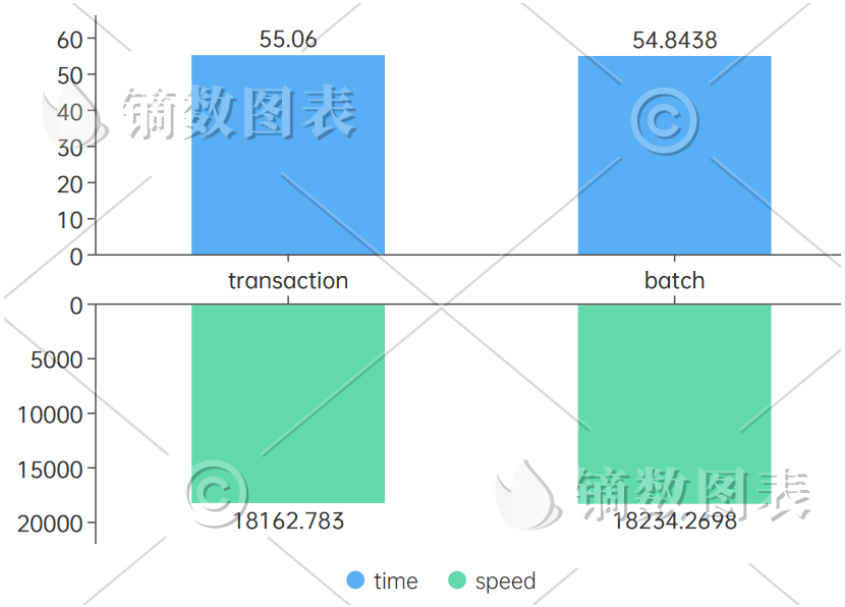
第二次测试我们导入数据量最大的 post_author_relation 表，数据量为 18746 条，剩下的四种方法每一种方法都测试五次（测试文件即附件中的 test2Connect_2.java 、test3Prepare_2.java、test4Transaction_2.java、test5Batch_2.java 文件），取平均值，统计图如下



由图可知，T 方法和 B 方法速度较快且相差不大，数据量有一定规模，结果较为精准，我们挑选这两种方法进行高级测试

高级测试

第三次测试我们先用附件中的 TestGenerator.java 文件生成一百万条数据量写入 test.txt 文件，这一百万条数据是符合 areas 表的（但去除了 post_id 的外键指向，若重新测试，需重新建表），我们两种方法都测试五次（测试文件即附件中的 test4Transaction_3.java、test5Batch_3.java 文件），取平均值，统计图如下



由图可知，导入一百万条数据，T 方法和 B 方法都用时 55 秒左右，B 方法略快

测试总结

由于我们本次 Project 数据量不大，所以本组采用 T 方法导入数据，若数据量非常大，采用 B 方法效率更快。同时本组测试为单线程，若多线程导入数据，能节省大量时间，不再多述

3. 数据准度

经测试，本组设计数据库数据符合测试文件，准确无误

第三部分：项目总结

本次项目通过建立 E-R 图确定各数据间的相关关系并设计创造各关联表，使用 Java 语言多种方式导入数据实现优化，最终完成数据库设计。在项目过程中，本组成员加深了对数据库、对 E-R 图、对数据导入操作的具体认知，也在协作中增强了团队意识，收获颇丰。