CS307 Project Part I Report

2.1 Group Information

成员及分工：

12410208 陈睿杰

Task 1: E-R Diagram

Task 2: Database Design

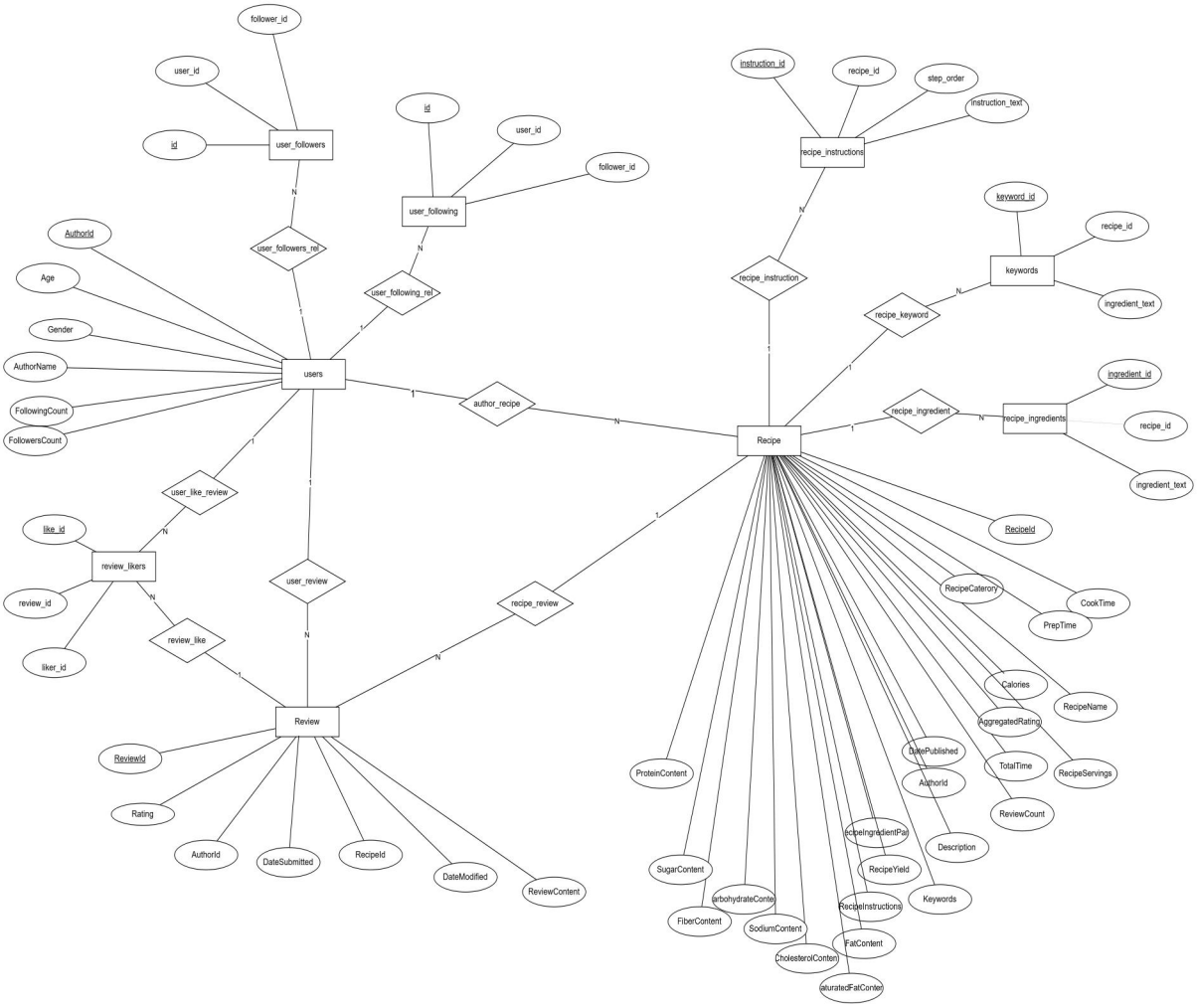
Task 4: File I/O

12412559 谢妍

Task 3: Data Import

Task 4: DBMS + bonus

2.2 Task 1　E-R Diagram

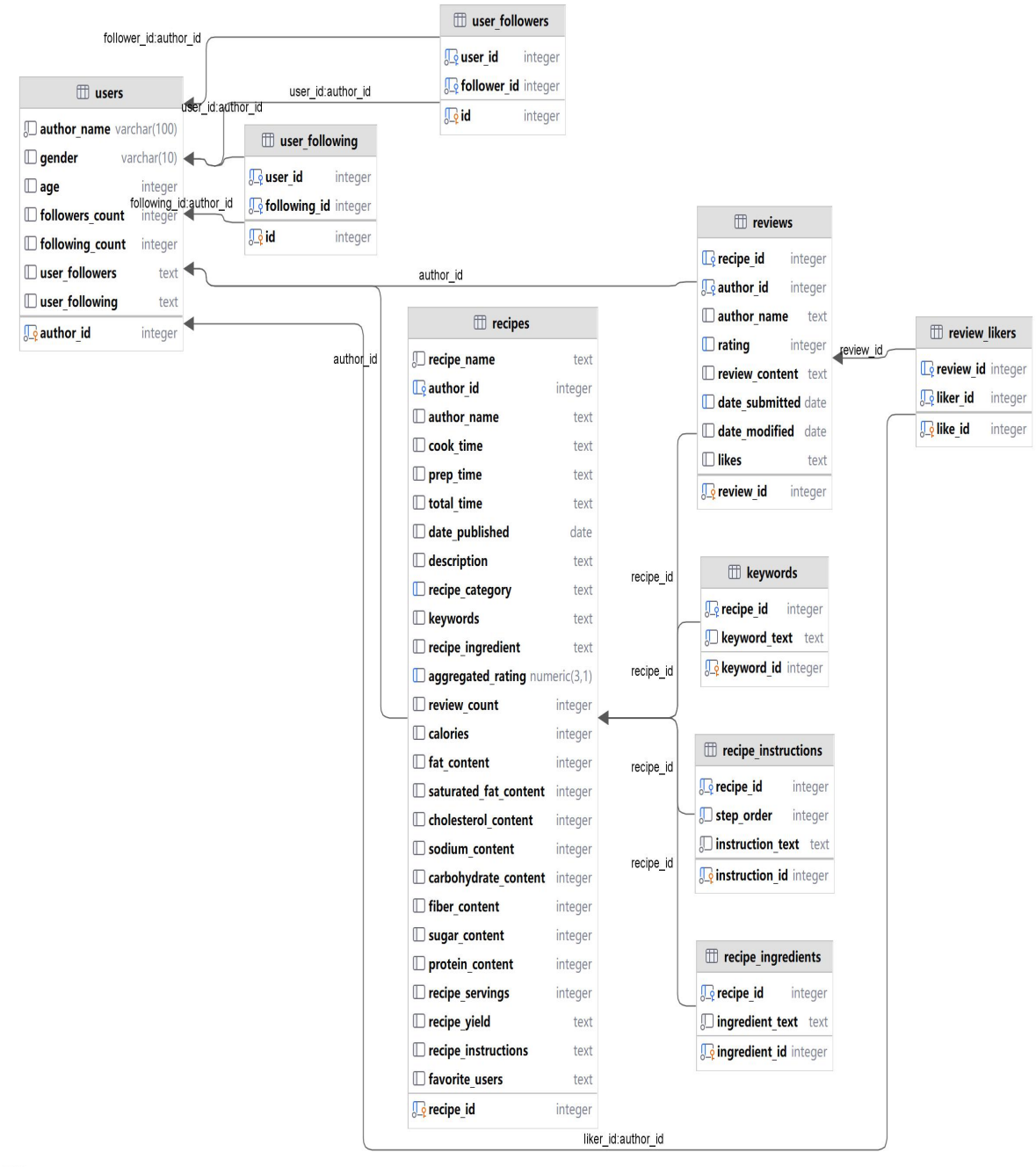


数据库的 E-R 图（使用 drawio）

2.3 Task 2　Database Design

一、数据库结构

我们设计的数据库完整结构如下：



datagrip 导出 diagram 图

二、表与字段设计说明

针对给定数据，我们设计了 3 个主表，6 个关联表，一共 9 个表来组织数据库，以下是每个表的详细信息。

1. users 表

含义：存储平台用户（食谱作者）的基本信息，是整个数据库的核心实体之一，用于关联食谱、评论、关注关系等模块。

字段说明：

author_id：主键（INT, NOT NULL），唯一标识每个用户，确保数据的唯一性。

author_name：用户姓名（VARCHAR (100), NOT NULL），记录用户的名称信息。

gender：性别（VARCHAR (10)），可选字段，记录用户性别。

age：年龄（INT），记录用户年龄。

followers_count：粉丝数（INT, DEFAULT 0），默认值为 0，记录用户的粉丝数量。

following_count：关注数（INT, DEFAULT 0），默认值为 0，记录用户关注的其他用户数量。

2. recipes 表

含义：存储食谱的核心信息，是平台的业务核心实体，关联作者、步骤、食材、关键词、评论等模块。

字段说明：

recipe_id：主键（INT, NOT NULL），唯一标识每个食谱。

recipe_name：食谱名称（TEXT, NOT NULL），记录食谱的名称。

author_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示该食谱的作者。

cook_time：烹饪时间（TEXT），记录食谱的烹饪时长。

prep_time：准备时间（TEXT），记录食谱的准备时长。

total_time：总时长（TEXT），记录食谱从准备到完成的总时长。

date_published：发布日期（DATE, NOT NULL），记录食谱的发布时间。

description：描述（TEXT），对食谱的详细说明。

recipe_category：分类（TEXT），食谱所属的分类，如“中餐”“西餐”等。

aggregated_rating：综合评分（NUMERIC (3,1)），记录食谱的综合评分，格式为“X.X”。

review_count：评论数（INT, DEFAULT 0），默认值为 0，记录该食谱的评论数量。

calories：卡路里（INT），食谱的卡路里含量。

fat_content：脂肪含量（INT），食谱的脂肪含量。

saturated_fat_content：饱和脂肪含量（INT），食谱的饱和脂肪含量。

cholesterol_content：胆固醇含量（INT），食谱的胆固醇含量。

sodium_content：钠含量（INT），食谱的钠含量。

carbohydrate_content：碳水化合物含量（INT），食谱的碳水化合物含量。

fiber_content：纤维含量（INT），食谱的纤维含量。

sugar_content：糖含量（INT），食谱的糖含量。

protein_content：蛋白质含量（INT），食谱的蛋白质含量。

recipe_servings：份数（INT），食谱可供应的份数。

recipe_yield：产量（TEXT），食谱的产出量描述。

favorite_users：收藏用户（TEXT），收藏该食谱的用户信息。

3. reviews 表
含义：存储用户对食谱的评论信息，建立用户与食谱之间的互动关联。
字段说明：
review_id：主键（INT, NOT NULL），唯一标识每条评论。
recipe_id：外键（INT, NOT NULL），关联 recipes 表的 recipe_id，表示评论对应的食谱。
author_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示发布评论的用户。
rating：评分（INT, NOT NULL），用户对食谱的评分。
review_content：评论内容（TEXT, NOT NULL），用户对食谱的评论文字。
date_submitted：提交日期（DATE, NOT NULL），评论的提交时间。
date_modified：修改日期（DATE），评论的修改时间（可选）。

4. recipe_instructions 表
含义：存储食谱的制作步骤，是食谱实体的详细扩展信息。
字段说明：
instruction_id：主键（INT, NOT NULL），唯一标识每个步骤。
recipe_id：外键（INT, NOT NULL），关联 recipes 表的 recipe_id，表示步骤对应的食谱。
step_order：步骤顺序（INT, NOT NULL），步骤在食谱制作中的顺序。
instruction_text：步骤文本（TEXT, NOT NULL），步骤的详细描述。

5. keywords 表
含义：存储食谱的关键词，用于食谱的检索和分类扩展。
字段说明：
keyword_id：主键（INT, NOT NULL），唯一标识每个关键词。
recipe_id：外键（INT, NOT NULL），关联 recipes 表的 recipe_id，表示关键词对应的食谱。
keyword_text：关键词文本（TEXT, NOT NULL），具体的关键词内容。

6. recipe_ingredients 表
含义：存储食谱的食材信息，是食谱实体的重要组成部分。
字段说明：
ingredient_id：主键（INT, NOT NULL），唯一标识每种食材。
recipe_id：外键（INT, NOT NULL），关联 recipes 表的 recipe_id，表示食材对应的食谱。
ingredient_text：食材文本（TEXT, NOT NULL），具体的食材内容。

7. user_following 表
含义：存储用户的关注关系，建立用户之间的社交关联。
字段说明：
id：主键（INT, NOT NULL），唯一标识每条关注关系。
user_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示发起关注的用户。
following_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示被关注的用户。

8. user_followers 表
含义：存储用户的被关注关系，从被关注角度体现用户之间的社交关联。
字段说明：
id：主键（INT, NOT NULL），唯一标识每条被关注关系。

user_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示被关注的用户。
follower_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示发起关注的用户。

9. review_likers 表
含义：存储用户对评论的点赞关系，建立用户与评论之间的互动关联。
字段说明：
like_id：主键（INT, NOT NULL），唯一标识每个点赞行为。
review_id：外键（INT, NOT NULL），关联 reviews 表的 review_id，表示点赞对应的评论。
liker_id：外键（INT, NOT NULL），关联 users 表的 author_id，表示发起点赞的用户。

2.4 Task 3　Data Import

一、数据导入脚本设计
本项目使用语言为 Java，基于 Marven 构建。我们利用 JDBC 接口编写了完整的、从 CSV 数据到完整数据库的完全自动导入脚本，项目完整结构如下：

```
CS307_Project1
├─── data
│    ├─── recipes.csv
│    ├─── reviews.csv
│    └─── user.csv
├─── src
│    └─── main
│         ├─── java
│              ├─── Task3
│                   ├─── Main.java
│                   ├─── Safety.java
│                   ├─── CSVReader.java
│                   ├─── UserImporter.java
│                   ├─── UserImporter_noPool.java
│                   ├─── RecipeImporter.java
│                   └─── ReviewImporter.java
│              └─── Task4
│                   ├─── MySQLTest.java
│                   ├─── FileIOTest.java
│                   └─── DBMSTest.java
│         └─── cpp
│              └─── C++Loader.cpp
└─── pom.xml
```

其中，C++Loader、MySQLTest、FileIOTest 及 DBMSTest 为第四部分测试使用。

三个.csv 原始文件都储存在 data 文件夹下，便于读取和查看。
六个核心类，功能分别为：

CSVReader：基于 OpenCSV 库的 CSV 文件读取工具，逐行读取原始.csv 文件并对数据中含有的符号（逗号，双引号，反斜杠等）进行基本处理。
Safety：一个数据库操作工具类，主要提供数据清洗、类型转换、安全处理和数据库管理功能，确保数据在插入数据库前的安全性和规范性。可以对整型、双精度浮点数、时间格式和多元素信息（如 recipes 表的 Ingredients 等）进行处理以及分割，确保数据在插入数据库前的安全性和规范性。、
UserImporter: 为用户数据导入类。功能包括创建用户 users 及其关联表 user_followers，导入用户数据，同时调用 Safety 方法以保证数据完整性与正确性。
RecipeImporter 为食谱数据导入类，功能包括创建食谱 recipes 及其关联表 keywords、recipe_ingredients、recipe_instructions，导入食谱数据，同时确保数据完整性与正确性。
ReviewImporter: 评论数据导入类，功能包括创建评论 reviews 及其关联表 review_likers，导入食谱数据，同时确保数据完整性与正确性。
Main 为主类，主要用于集中调用各种方法。

二、脚本导入流程

脚本使用 OpenCSV 对 CSV 文件进行解析和读取，处理 CSV 数据的特殊情况（如引号、分隔符等），最后转换为 Java 数据结构（List<String[]>）。接着使用 JDBC 进行数据写入，主要负责数据库连接和操作、执行 SQL 语句插入数据和管理数据库事务和连接功能。

（1）主表及关联表的创建

首先依次创建 users 主表、截止到 users_followers 和 users_following 两关联表，导入 users 数据并建立相应外键关联后，创建 recipes 主表及 keywords、recipe_ingredients、recipe_instructions 三个关联表，导入 recipes 数据，最后创建 reviews 及其关联表 review_likers，导入 reviews 数据。
注意：recipes 和 reviews 中存在对于 user_id 和 recipe_id 的外键约束，因此数据必须严格按照 users → recipes → reviews 的顺序导入。

（2）导入数据

导入数据时，对数据分区处理，将数据按 1000 条每批进行分区，并使用 6 个线程并行处理不同数据，再利用 PreparedStatement 进行批量数据库操作，导入完成后统计各表记录数量，完成数据统计。同时，对于格式错误的数据，我们进行记录和输出，便于后续修正。
其中，各主表数据记录数如下:
users 数据记录数为 299892，recipes 数据记录数为 463562，（有效行 369771），reviews 记录数为 533026（有效行 472757）。
各关联表数据记录数如下：
user_followers 数据记录数为 1549415，user_ following 数据记录数为 743968。
keywords 数据记录数为 1256956，ingredients 数据记录数为 2063029，instructions 数据记录数为 765005。
review_likers 数据记录数为 1192178。

脚本使用 JDBC 实现了完全自动化：输入 CSV 文件路径，输出完整构建的数据库表，包含数

据清洗、转换、验证全流程。

三、进阶部分

测试环境：
联想拯救者 Y7000P
CPU：Intel i7-14700HX 8 核 16 线程
内存：16GB
磁盘：SSD 1 TB
软件配置
操作系统：Windows 11 64-bit
编程语言：Java 22
开发环境：IntelliJ IDEA 2024.2.1、DataGrip2025.2.2
依赖：PostgresSql 42.6.0、OpenCSV 5.8
数据库: PostgreSQL 17.6
数据集: 约 30 万条 Users 记录

测试方式：清空数据库，导入 Users 表的所有数据，记录开始时间与结束时间，相减得到时间开销。分别使用并行批量导入、多线程单独导入和单线程批量导入三种导入方式，每种方法均单独测试三次，记录导入时间，进行分析比较。

一、数据导入方法说明

方法一：**并行批量**导入
实现方式：6 线程并行处理，每 1000 条 SQL 统一提交
技术特点：多线程、批量插入
优势：充分利用多核 CPU，减少数据库连接开销
适用场景：大数据量、高性能要求的场景

方法二：**并行批量+连接池**导入
实现方式：6 线程并行处理，连接池复用，每 1000 条 SQL 统一提交
技术特点：多线程、批量插入，每个线程复用连接
优势：每个批次在独立事务中执行，确保数据一致性

方法三：**多线程单独**导入
实现方式：批处理规模设置为 1
技术特点：多线程顺序处理，单独提交
优势：实现简单，利用 CPU 多线程
劣势：每次都需与数据库通信，资源占用率高

方法四：**单线程批量**导入
实现方法：线程数（THREAD_COUNT）设置为 1
技术特点：单线程处理,每 1000 条 SQL 统一提交
优势：数据库驱动优化，减少网络往返
劣势：内存占用较高

二、时间对比结果

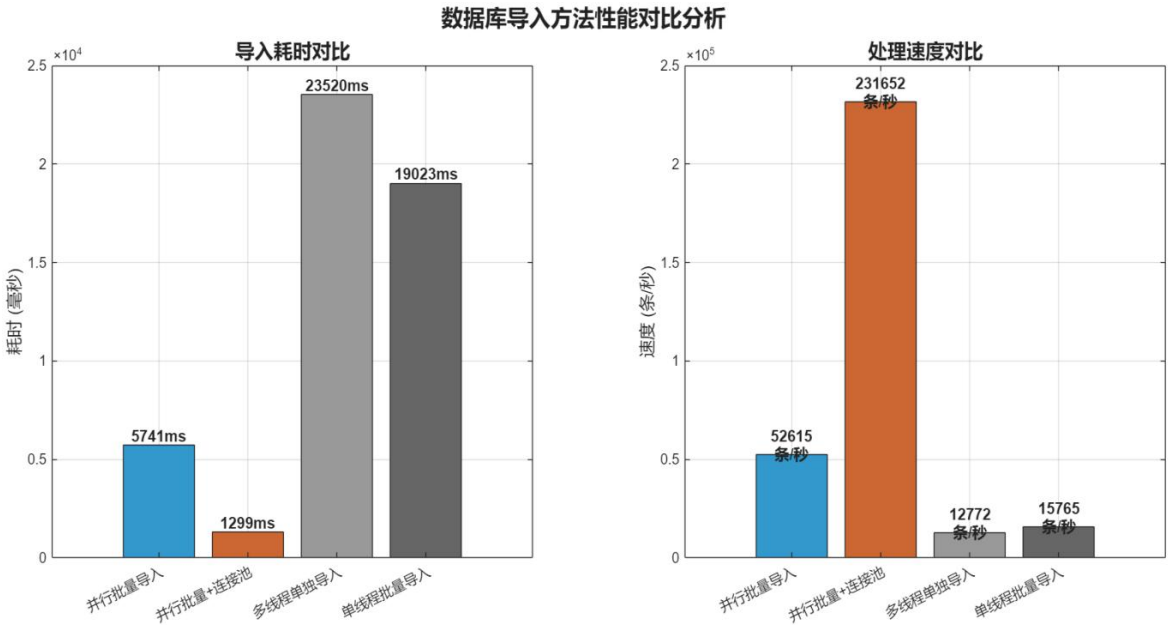
并行批量导入（方法一）：
第一次测试
总耗时: 6446 ms 平均速度: 46523 条/秒
第二次测试
总耗时: 5357 ms 平均速度: 55981 条/秒
第三次测试
总耗时: 5419 ms 平均速度: 55340 条/秒

并行批量+连接池导入（方法二）：
第一次测试
总耗时: 1395 ms 平均速度:214976 条/秒
第二次测试
总耗时: 1284 ms 平均速度: 233560 条/秒
第三次测试
总耗时:1217 ms 平均速度: 246419 条/秒

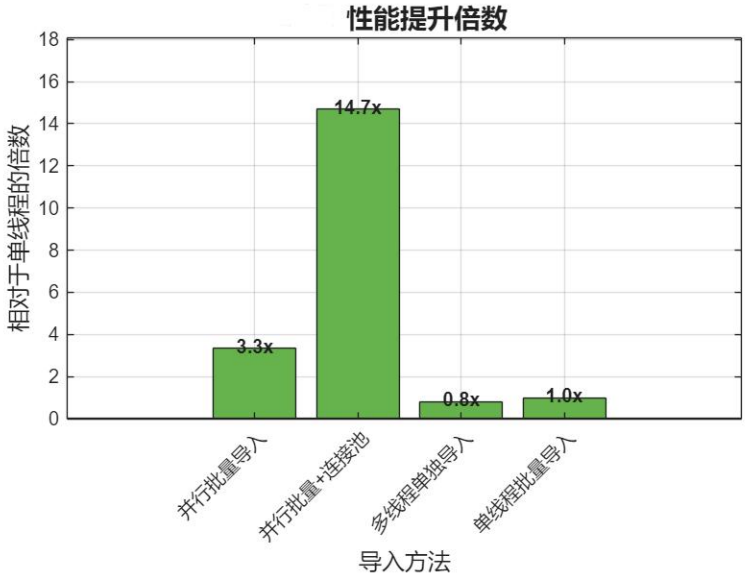
多线程单独导入（方法三）
第一次测试
总耗时: 22916 ms 平均速度: 13086 条/秒
第二次测试
总耗时: 22723 ms 平均速度: 13197 条/秒
第三次测试
总耗时: 24921 ms 平均速度: 12033 条/秒

单线程批量导入（方法四）
第一次测试
总耗时: 18819 ms 平均速度: 15935 条/秒
第二次测试
总耗时: 19010 ms 平均速度: 15775 条/秒
第三次测试
总耗时: 19240 ms 平均速度: 15586 条/秒

利用 matlab 进行可视化：



性能对比：



三、实验结果分析

根据以上结果，不难看出，虽然性能有波动，但并行批量导入的效率高于其他两种导入方式，而加入连接池进一步优化了导入速度。我们采用以下脚本以对导入效率进行优化：

并行处理：从单线程改为 6 线程并行，提高了 CPU 利用率，充分利用多核处理器。

批量操作：每 1000 条记录批量提交，批量提交减少了事务提交次数，提高了数据库处理效率，同时避免了内存溢出。

使用连接池：每个线程复用连接，每个批次在独立事务中执行，确保数据一致性

优化后的脚本在保持数据一致性的前提下，大幅缩短了导入时间，特别适合处理大规模数据集。该脚本满足了数据导入任务的要求，实现原始 csv 数据 →完整、安全、可以导入数据库的数据的自动转化，并在性能优化方面达到了较高水平。

2.5 Task 4　Compare DBMS with File I/O

一、DBMS 测试

（1）环境：同上，不再赘述。

（2）测试方式

利用现有 users 表，在第三问的基础上增加对查询、更新，删除和复杂聚合查询、连接查询效率的测试。其中，查询分别测试了全表查询，年龄 25-35 岁的用户，粉丝数>100、女性用户，更新测试了 30 岁以下用户粉丝+1，删除测试删除了所有年龄大于 50 岁的用户。由于我们直接在原 users 表进行测试，为了保护数据，在进行删除和更新测试时，我们创建了一个临时副本，以保持数据的完整性。同时，为了避免外键约束造成的不合法操作，测试时数据库 user_follower 表和 user_following 表应暂时删去外键约束条件。

（3）测试结果

DBMS 性能测试结果（现有users表）

操作类型	记录数	执行时间(ms)
全表查询	299,872	287
年龄条件查询(25-35)	78,219	96
粉丝数条件查询(>100)	1,930	116
性别条件查询(Female)	135,327	96
更新操作	85,144	360
删除操作	10	46
复杂聚合查询	-	123
连接查询	-	2,039

二、File I/O 测试

（1）环境:

联想拯救者 Y9000P

CPU: Intel i9-14900HX

内存：32 GB

磁盘：SSD 1 TB

软件配置

操作系统：Windows 11 64-bit

编程语言：Java 23

开发环境：IntelliJ IDEA Community Edition 2024.3.3

依赖库：OpenCSV 5.9、Apache POI 5.2.3
数据集: 约 30 万条 Users 记录

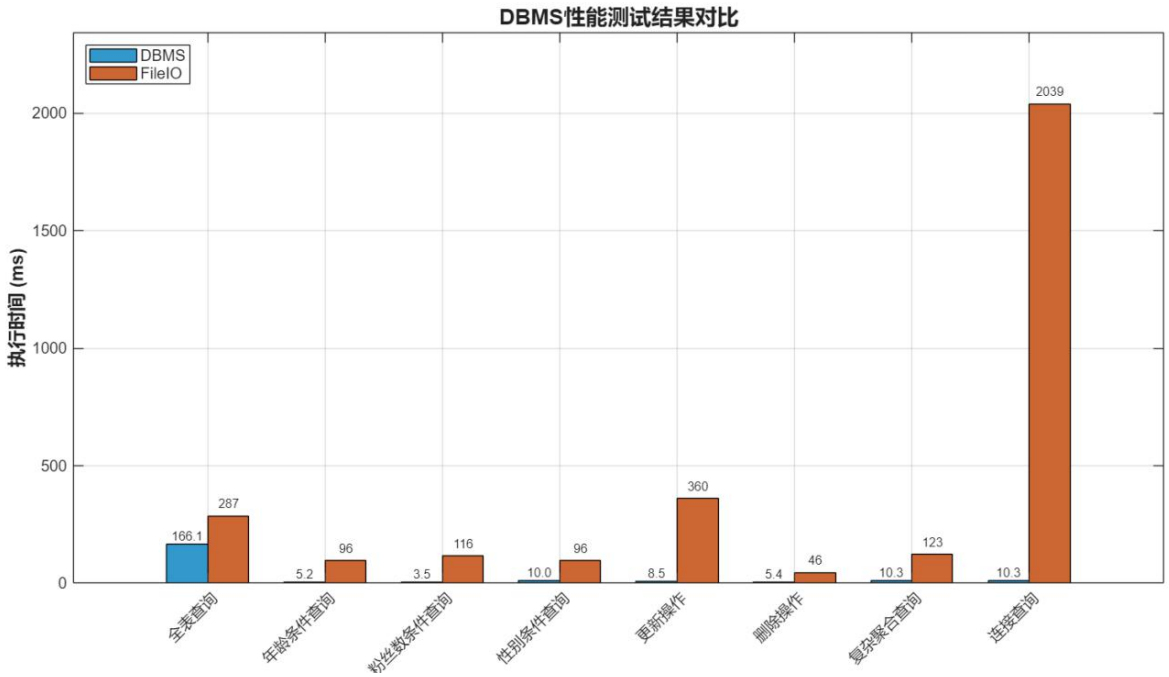
（2）测试方式
与 DBMS 测试操作相同，每个操作执行 10 次，记录每次耗时并计算平均耗时。通过 Apache POI 输出到 Excel 表格，再用 matlab 进行整合。

（3）测试结果

性能测试平均时间结果

操作类型	平均时间(ms)
全表读取	166.108920
全表写入	48.707330
查询_年龄25-35	5.164160
查询_粉丝数100	3.445770
查询_女性用户	10.044920
更新_30岁以下	8.518760
删除_50岁以上	5.370480
复杂聚合查询	10.348880
复杂连接模拟	10.275880

三、时间开销对比



不难看到，DBMS 在各操作的时间开销均优秀，远超文件 IO 读写。操作高效、稳定，适合用于复杂数据的存储，查询和修改。

对于 File I/O 与数据库读写的性能表现，我们进行分析：

File I/O 耗时更长：解析 CSV 并创建 Java 对象需要额外的 CPU 计算和内存开销（如字符串分割、类型转换、对象实例化）。对于 30 万条数据量，文件 I/O 逐行查询（无索引）耗时几乎接近数据库的索引查询，这可能是因为 FileIO 直接顺序读取内存中的数据，而无需进行数据库通信、SQL 解析等操作，但数据库的查询优化（如 B 树索引）在大数据量下优势会呈指数级放大。我们可以得到结论：File I/O 适用与文件 I/O 适合小数据量的数据交换（如 CSV 导出/导入），而数据库适合长期存储、复杂查询、事务性操作的业务场景。

四、进阶部分

进阶部分中，我们分别探究了使用 C ++ 编写脚本和 MySQL 对导入速度的影响。

（1）在以上的性能测试中，我们对于全表查询进行了测试，显然在数十万条的同时查询下，我们的脚本仍然运行效率良好，能够处理高并发的指令。

（2）使用 C ++ 编写导入脚本

我们知道 C ++ 是一个注重性能的语言，在同样情况下，使用 C ++ 编写的程序往往速度更快，内存控制更精确，减少了很多不必要的开销。因此，我们使用 C ++ 编写了同样具有批处理和多线程导入的脚本，在同样情况下（批规模 1000，6 线程）对比 Java 脚本的导入速度，以期研究编程语言对导入速度的影响。

同样的，我们生成 sql 语句，利用 libpq 库（PostgreSQL 官方 C 语言客户端库）分批多线程执行，记录开始以及结束时间，得到导入速度。为了使导入效率的可能差异更明显，我们选择 Task 3 中的版本一（**并行批量导入，无连接池**）。

测试环境：仍旧同 Part3。
开发环境：VS Code 1.106.0
编程语言：C ++ 17
编译器：minGW-w64

注意：由于我们使用的 MinGW-w64 为 64 位，和 PostgreSQL 第三方库（仅支持 32 位）二进制格式不兼容，因此该脚本的编译和运行将在 MSYS2 MINGW64 上进行。

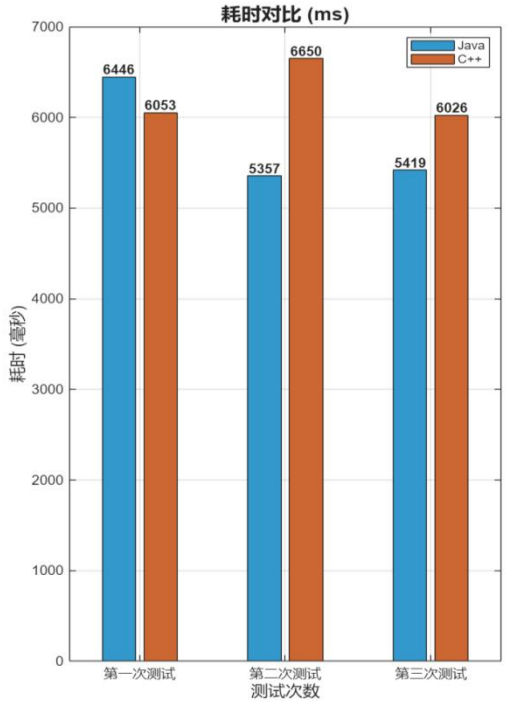
测试结果

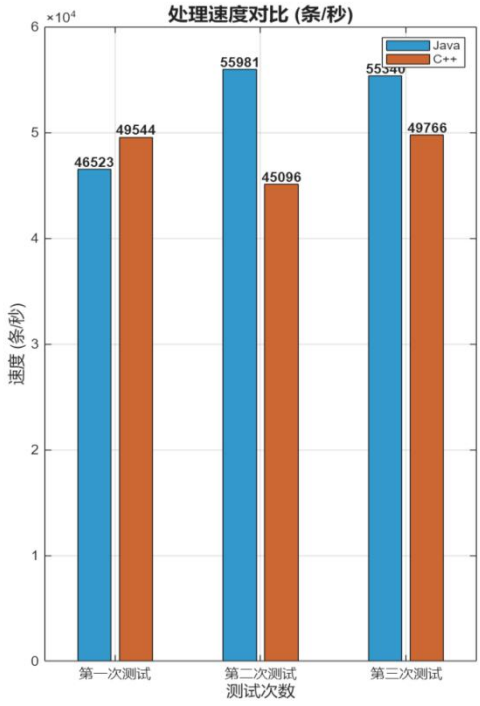
第一次测试
总耗时：6053 ms 平均速度 49544 条/秒

第二次测试
总耗时：6650 ms 平均速度 45096 条/秒

第三次测试
总耗时：6026 ms 平均速度 49766 条/秒

不难看到 C ++ 脚本的导入耗时相当稳定，对比 Java 脚本：





速度并无明显优势，甚至比 Java 脚本更慢。推测原因：
对于 IO 密集型任务，编程语言影响有限，因为时间开销大多在与数据库进行连接通信上。如果想要对导入速度进行进一步优化，应该针对连接提交等操作优化（如批提交、建立连接池等），而非更换编程语言。

在编写脚本过程中，明显感受到 C ++的语句较 Java 相当简洁，但是各种三方库（如 Postgres）的支持并不好，不太适合用于大型、复杂项目的开发。相比之下，Java 的各种库和接口都十分成熟，编写和调试都更为简单。

（3）使用 MySQL 测试

除 PostgresSQL 外，MySQL 是另一个被广泛使用的数据库。两者主要的区别是，PostgreSQL 是一个对象关系数据库管理系统，支持许多 Web 应用程序、动态网站和嵌入式系统，而 MySQL 是一个关系数据库管理系统，更适用于创建用户较少的内部应用程序，或者创建具有更多读取次数和较少数据更新的信息存储引擎。

同样，我们利用并行批量 + 连接池优化的 Java 脚本，对 MySQL 数据导入效率进行测试。

测试环境：依然同上！
版本：MySQL 8
注意：MySQL 在 8.0.16 版本之前不支持标准的 CHECK 约束

测试结果

第一次测试

总耗时: 5879 ms 平均速度: 51010 条/秒

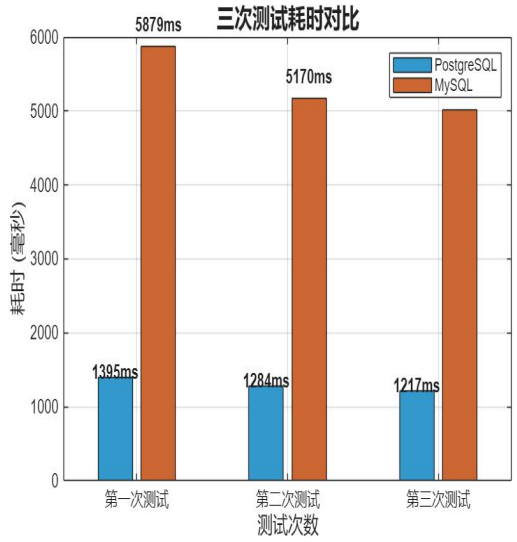
第二次测试

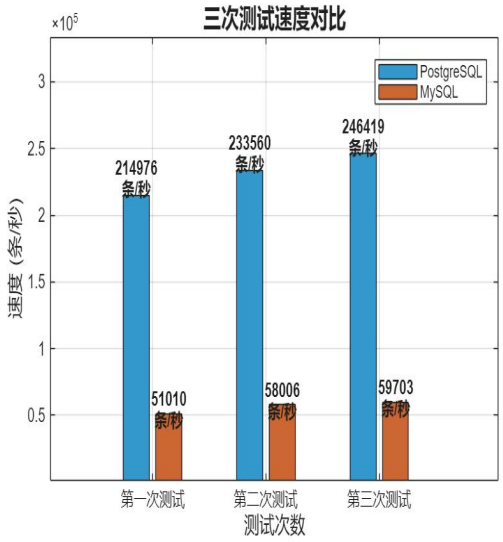
总耗时: 5170 ms 平均速度: 58006 条/秒

第三次测试

总耗时: 5023 ms 平均速度: 59703 条/秒

与 Postgres 对比：





可以看到 Postgres 在导入速度上有绝对优势。经过资料查找，我们还发现 MySQL 在复杂查询、修改等操作效率也不如 Postgres，但胜在其体量小，上手难度低，且工具生态较为完善（如 Percona Toolkit、sys Schema），云服务支持好（AWS，Aurora，Google，Alibaba 等都支持云 MySQL 服务），对于中小型开发者是一个很好的选择。