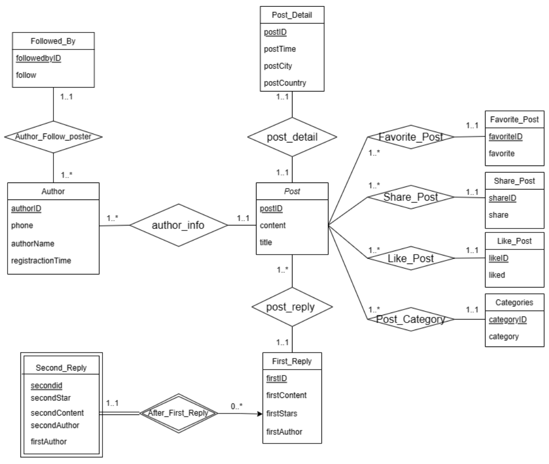
project 1

Spring 2023 CS307 Project Part1
number: 224

- 黄硕 12110810
  50% Improve the extraction. Improve the design. Import data into the file, data grip. Optimize the algorithm.
- 刘晓群 12110943
  50% Draw ER diagram. Design the database. Extract data from json to csv. Write descriptions. Import data into the file, data grip.

Task 1: E-R Diagram

The site diagrams.net is used to make the E-R diagram.



Task 2: Database Design

Clarification

1. All data items base on two files `posts.json` and `replies.json` .
2. The design strictly follow the requirements of the three normal forms.
3. We use a primary key and foreign keys to indicate important attributes and relationships about data, shown in the ER diagram.
4. Every row in each table can be uniquely identified by its primary key. Shown in the ER diagram. The primary key in the rectangles is underlined.
5. Every table is involved in a foreign key. The straight lines between them shows their relationships.
6. There are no circular foreign-key links in the ER diagram.

7. From data analysis, we found that all the tables has no column with null keys respectively, which means all "Not Null" column in this design.
8. Some column is constrained with the system-generated self-increment ID column, but other columns are constrained with the a "unique" constraint. (By later announcement on the features)
9. All data type is correct.

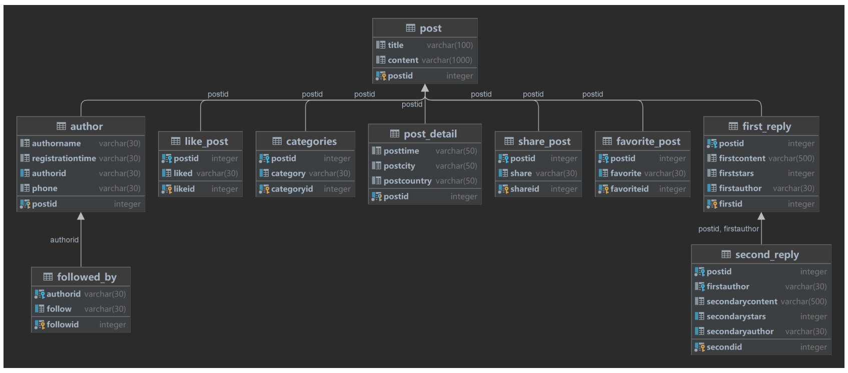
The element in posts. json:

Post ID : The id of post, **unique** --integer
Title : The title of post --string
Category : The category of post --string array
Content : The content of post --string including ","
Posting Time : The time of posting. Posting Time is later than the author registration time --string
Posting City : The city of posting. The city is randomly generated and it has no relation to the location of author. --string
Author : The Author of post, unique --string
Author Registration Time : The registration time of author. Must be earlier than the Posting Time. --string
Author's ID : ID of author. The ID verification code is valid, but does not verify that the birth time is earlier than the registration time --string
Author's Phone : phone of author --string
Authors Followed By : Accounts who followed by author. The authors in this field may not appear in Author field. --string array
Authors Who Favorited the Post : Accounts who favorited the post. The authors in this field may not appear in Author field. --string array
Authors Who Shared the Post : Accounts who shared the post. The authors in this field may not appear in Author field. --string array
Authors Who Liked the Post : Accounts who like the post. The authors in this field may not appear in Author field. --string array

The element in replies. json:

Post ID : The id of post in post.josn, unique --integer
Reply Content : the content of reply --string
Reply Stars : the star of reply --integer
Reply Author : the author of reply. The authors in this field may not appear in Author field. --string array
Secondary Reply Content : the content of sub reply --string
Secondary Reply Stars : the star count of sub reply --integer
Secondary Reply Author : the author of sub reply. The authors in this field may not appear in Author field. --string array
10. The design is easily expand is there are more data. Shown below.

E-R diagram



Descriptions

POST

POST is the main table, which contains main structure of a post.

- `Post_ID` : The given unique order in posts.json to identify each post.It is the primary key of the table.
  The `postID` continues to function as identifying details of the post stored in the foreign key "`postID`", which is related to **Post_detail** table.
  The `postID` which identify different authors is stored in the foreign key "`postID`", which is related to **Author** table.
  The `postID` which identify different first replys is stored in the foreign key "postID", which is related to **First_Reply** table.
  The `postID` continues to function as identifying different characteristic about authors relevant of the post(Followed by, favorite, share or like the post) is stored in the foreign key "`postID`", which is related to **Favorite_Post**, **Share_Post**, **Like_Post**, **Categories** table.
- `Content` : Show the content of the post .
- `Title` : Show the title of the post.

POST_DETAIL

POST_DETAIL is built for storing more information about the post, about when and where it's posted.

- `postID` : The primary key in this table, identify each information. The `postID` which identify different details is stored in the foreign key "`postID`", which is related to **Post** table.
- `Posting_Time` : Describe the time it was posted. Must be later than the `Registraction Time` in the **Author** table.
- `Posting_City` : The city where the post is posted.
- `Posting_Country` : The city belongs to .

AUTHOR

More information about the author who post the post.

- `authorID` : Uniquely identifies each author.
- `phone` : The phone number of the author.
- `authorName` : The name of the author.

- `registrationTime`: The time when the author registered.
- `postID`: The ID of the post(s) that the author has published.It is not null and the primary key of this table. The `postID` which identify different authors is stored in the foreign key "`postID`", which is related to **Post** table.

Followed_By

The people who follow the author which used as a addition of the author's innformation.

- `followID`: The self-increment number which identify different followers, using as the primary key.
- `follow`: The authors who follow.
- `authorID`: The ID of the author who is followed by others.It is not null and used as a foreign key related to **Author** table.
- To make the unique column,using `authorID` and `follow` from the composite unique column.

FIRST_REPLY

The **First_Reply** table contains information about the first reply to each post.

- `firstID`:The self-increment number,using as the primary key.
- `firstContent`: The content of the first reply.
- `firstStar`: The number of stars given to the first reply.
- `firstAuthor`: The author of the first reply.
- `postID`: The ID of the post to which the first reply is responding. The `postID` which identify different first reply is stored in the foreign key "`postID`", which is related to **Post** table.
- To make the unique column,using `postID` and `firstAuthor` from the composite unique column.

SECOND_REPLY

The **Second_Reply** table contains information about the second reply to each post.

- `secondID`: The self-increment number,using as the primary key.

-
  ```
   `firstAuthor`:To show the reply belong to the first_reply's poster.
  ```
- `secondaryContent`: The content of the second reply.
- `secondaryStars`: The number of stars given to the second reply.
- `secondaryAuthor`: The author of the second reply.
- `postID` :To show the reply belong to which post.
- To make the unique column,using `postID`, `secondaryAuthor` and `secondaryStars` from the composite unique column.
  -using the combination of `postID` and `firstAuthor` as the composite foreign key to relate the combination of `postID` and `firstAuthor` in **First_Reply**.

Categories

Contains information about which category the post belongs to.

- `postID`: It is not null and the foreign key related to the **Post** table.
- `categoryID`: The self-increment number,using as the primary key.
- `category`: The post's category.
- To make the unique column,using `postID` and `category` from the composite unique column.

Favorite_Post

Contains information about each post's interactions with users, including who favorites the post.

- - `favoriteID`: The self-increment number,using as the primary key.
- `postID`: It is not null and the foreign key related to the **Post** table.
- `favoriteID`: The id to identify each authors who favorited the post.
- `favorite`: The authors who favorited the post.
- To make the unique column,using `postID` and `favorite` from the composite unique column.

Share_Post

Contains information about each post's interactions with users, including who shares the post.

- `shareID`: The self-increment number,using as the primary key.
- `postID`: It is not null and the foreign key related to the **Post** table.
- `shareID`: The id to identify each authors who shared the post.
- `share`: The authors who shared the post.
- To make the unique column,using `postID` and `share` from the composite unique column.

Like_Post

Contains information about each post's interactions with users, including who likes the post.

- `likeID`: The self-increment number,using as the primary key.
- `postID`: It is not null and the foreign key related to the **Post** table.
- `likeID`: The id to identify each authors who liked the post.
- `liked`: The authors who liked the post.
- To make the unique column,using `postID` and `liked` from the composite unique column.

An SQL file `console_1.sql` is in the compressed zip as an attachment that contains the DDLs (create table statements) for all the tables.

Task 3: Data Import

Description of test environment and the PostgreSQL version

Test environment (Both with 256GB SSD)

```
Windows: 处理器	AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz 机带 RAM	16.0 GB (13.9 GB 可用) 系统类型	64 位操作系统, 基于 x64 的处理器
MAC: 型号标识符: MacBookPro16,1  处理器名称: 六核Intel Core i7  处理器速度: 2.6 GHz  处理器数目: 1  核总数: 6  L2缓存（每个核）: 256 KB  L3缓存: 12 MB  超线程技术: 已启用  内存: 16 GB
```

Language version

```
Python 3.10.2
java 16.0.2 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)
```

Task 3.1 Basic Requirements: 10%

Description

Do the following steps for `posts.json` and `replies.json` respectively.

1. Set the DBMS host, database name, username and password in the script.
2. Put the csv file to be imported into the directory where the Java script is located using `main.py` and `main2.py` into `data.csv` and `data2.csv`.
3. Then the script will read the file one line at a time, which includes several columns of data. Due to the comma in the content, we convert the "," in the csv into ";" to cut the potential problems using `convert.py`.
4. Choose multiply columns in the `data.csv` and `data2.csv` using `select_col.py`.
5. Separate different value in the csv into different columns in the new csv file using `divide_val.py`.
6. There maybe one First-Reply consistent with more than one Second-Reply, so there will be duplicate rows in the `First_Reply.csv`. We solve it by using `changeFirst` to remove the additional rows.
7. Separate all data into the previous tables, and load them to the data grip, the self-increment sequences are automatically generated.

Import script

Use the import of Favorite_Post as an example.

```java
public static void main(String[] args) {
        Properties prop = loadDBUser();
        List<String> lines = loadCSVFile();    //load the data in the csv
        // Empty target table
        openDB(prop);
        clearDataInTable_favor();
        closeDB();
        int cnt = 0;
        //start timing
        long start = System.currentTimeMillis();
        openDB(prop);  //openDB
        setPrepareStatement_favor();
        try {
            for (String line : lines) {
                if (line.startsWith("PostID"))
                    continue; // skip the first line
                loadData_favor(line);//do insert command
                if (cnt % BATCH_SIZE == 0) {
                    stmt.executeBatch();
                    System.out.println("insert " + BATCH_SIZE + " data successfully!");
                    stmt.clearBatch();
                }
                cnt++;
            }
            if (cnt % BATCH_SIZE != 0) {
                stmt.executeBatch();
                System.out.println("insert " + cnt % BATCH_SIZE + " data successfully!");
            }
            con.commit();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        closeDB(); //close the database
        long end = System.currentTimeMillis();   //stop timing
        System.out.println(cnt + " records successfully loaded");
        System.out.println("Time consuming :  " + (end-start/1000) + "s");
        System.out.println("Loading speed : " + (cnt * 1000L) / (end - start) + " records/s");
    }
}
```

Cautions

- There are spaces in the imported data, we do the data cleaning during the loading process.
- If your host, dbname, username and password are not set correctly, you will get the caution "Database connection failed", please check the parameters again.
- If there are space in the json, there will be a space in the database either.
- The city and the country are bonded in the **Author** table, we separate it into two parts using `deduplicate` and `dividecity`.

Task 3.2 Advanced requirements: 10%

There are five methods using in comparison to find which one is the most effective.
We use the load of table **Second_Reply** to demonstrate the difference between different algorithms.

More ways to import data

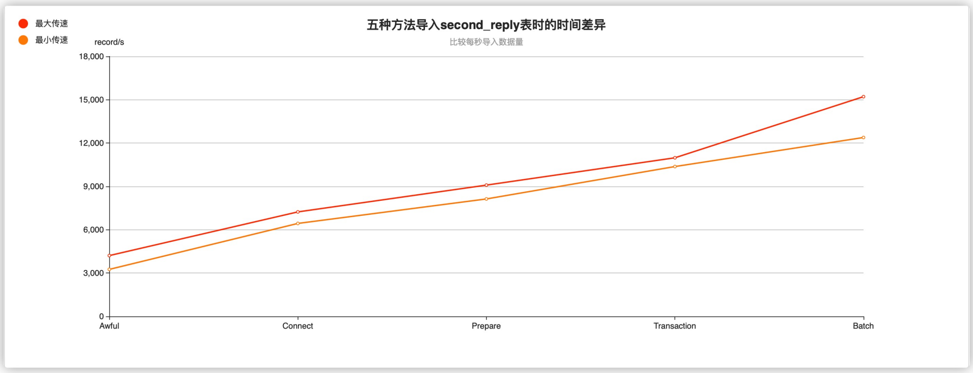
The use of prepared statements has the following benefits:
Improving performance: Preprocessing statements only need to be parsed once and can be reu sed, reducing the number of times the database parses the statements, thereby improving perf ormance.
Preventing SQL injection: Preprocessing statements separate SQL statements and parameters,

and parameters are passed to the database through parameter binding, effectively preventing SQL injection attacks.
Easy maintenance: Using preprocessed statements can separate SQL statements from paramet ers, making the reuse and maintenance of SQL statements more convenient.
Improving security: Using preprocessed statements can effectively reduce the risk of SQL inject ion attacks, thereby improving database security.
In summary, using preprocessed statements can improve the performance, security, and mainta inability of databases, and is a recommended database operation method.

Comparative Analysis

Computational efficiencies between these ways.

The polygon



The later four ways all improve the import process.

More of the test environment

We run the overall loading efficiency using the quickest method on different devices and the time costs shown as follows.
Windows:

```
431 records successfully loaded
Time consuming :  1.209s
Loading speed : 356 records/s
```

Mac os:

```
431 records successfully loaded
Time consuming :  1.300s
Loading speed : 316 records/s
```

Windows shows better in importing the data.

Task 3.3 Data Accuracy checking: 10%

All requirements met.
Tested in class.