数据库概论第一次实习作业

原梓轩 2200010825
陈润璘 2200010848
任子博 2200010626

本次实习的目标是设计代码评测网站的数据库，包括列举业务需求、设计ER图、将ER图转换为关系表、用SQL语句实现业务功能。

一、业务需求

代码评测网站以代码提交和评测为主要功能，在此之外，我们还实现了发布帖子和评论的功能。

- **代码提交**:
  用户可以针对题目提交代码，提交的代码会被上传数据库，以供后续评测。
- **代码评测**:
  用户提交代码后，系统从数据库获取时间空间限制和测试用例并进行评测，评测结果会被上传数据库并可供用户查询。
- **论坛**:
  用户可以针对问题发布帖子，帖子会被上传数据库。用户也可以对已发布帖子进行评论。

二、ER图设计

根据上述业务需求，确定实体及实体间的联系

1. 实体

共设立六种实体，并确定它们各自的属性（主码用下划线来标识）

- **用户(User)**
  （用户id，用户名，密码，等级）
- **题目(Problem)**
  （题目id，题目名，作者id，发布时间，题目描述，难度，时间限制，内存限制）
- **提交(Submission)**
  （提交id，用户id，题目id，提交时间，代码，编程语言，状态，分数）
- **帖子(Post)**
  （帖子id，标题，作者id，问题id，发布时间，内容）
- **评论(Comment)**
  （评论id，帖子id，评论者id，评论时间，内容）

以上五种均为强实体，而对于输入输出对，它依赖于题目，因此将其设置为弱实体。

- **输入输出对(IOpair)**
  （题目id，输入输出对id，输入，输出，得分，类型）

2. 联系

实体之间存在如下的联系：

由用户发出的一对多联系：

- **出题**：用户-题目之间的一对多联系。一个题目只能有一个出题人。
- **写入**：用户-提交之间的一对多联系。一次提交只能由一个用户发出。
- **发帖**：用户-帖子之间的一对多联系。一个帖子只能由一个用户编写。
- **发表评论**：用户-评论之间的一对多联系。一个评论只能由一个用户发表。

题目评测过程中发生的联系：

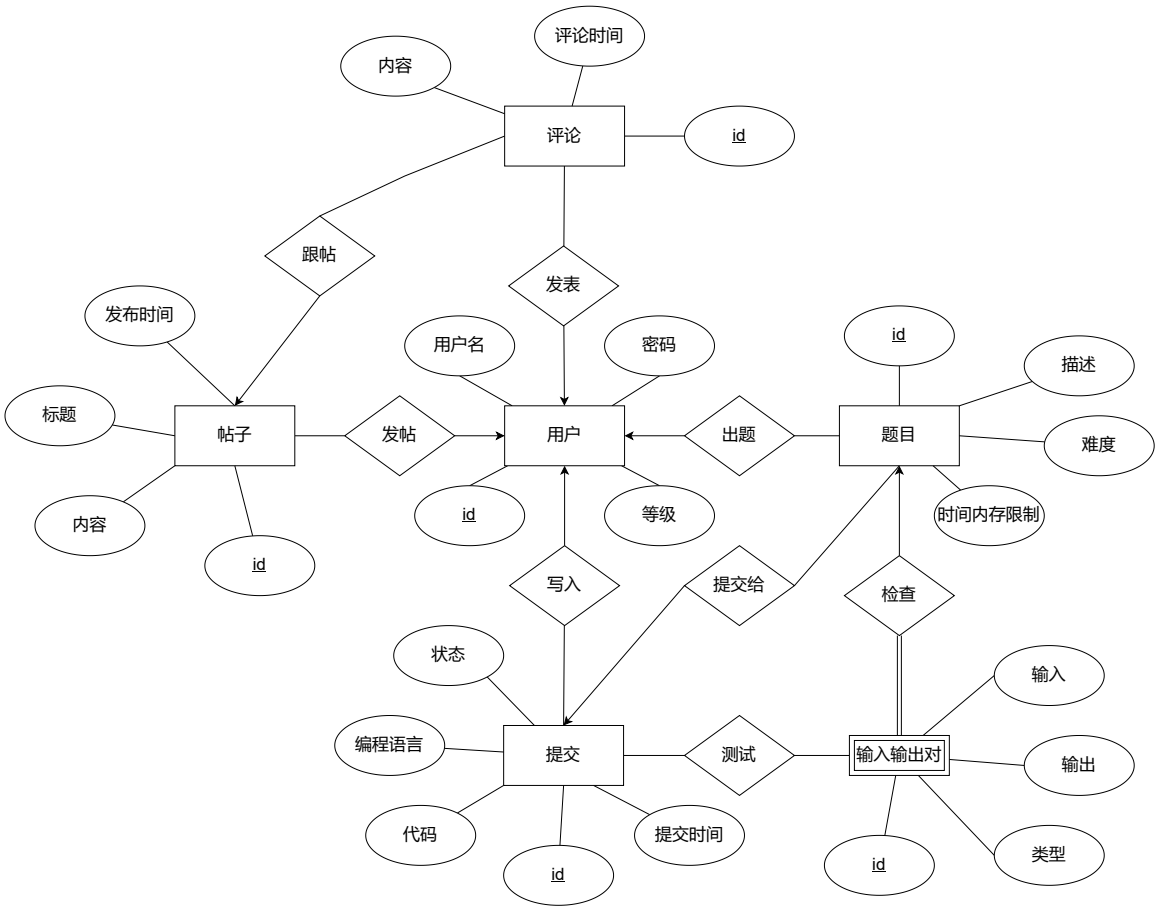
- **提交给**：题目-提交之间的一对多联系。提交被传给特定题目进行评测。
- **检查**：题目-输入输出对之间的一对多联系。输入输出对是对提交正确性进行评测的依据。
- **测试点**：输入输出对-提交之间的多对多联系。一个提交可以被多个输入输出对评测，一个输入输出对可以评测多个提交。

论坛中发生的联系：

- **跟帖**：帖子-评论之间的一对多联系。一个帖子下可以有多条评论，一条评论只能属于一个帖子。

3. ER图

根据列出的实体和联系，绘制ER图如下：



三、关系表创建

完成ER图设计后，我们将ER图转换为关系表。

6个实体各自对应一张表，弱实体所对应的表中有外键指向其依赖的强实体。测试点(Checkpoint)作为多对多联系，需要新建一张表来存储。其余联系均为一对多联系，不需要单独创建表，而是在"多"的一方添加外键指向"一"的一方。

四、数据库建立

1. 基本数据库准备

数据库链接详见 配置文件 。

```
import json
import pymysql
from IPython import get_ipython

pymysql.install_as_MySQLdb()

with open('./config.json') as f:
    config = json.load(f)
db = pymysql.connect(**config)
cursor = db.cursor()
db.commit()

ip = get_ipython()
ip.run_line_magic('load_ext', 'sql')
ip.run_line_magic('sql', f"mysql://{config['user']}:{config['password']}@{config['host']}:{config['port']}")
ip.run_line_magic('sql', f"use {config['db']};")
```

清理先前的数据，建立新的数据库

```
%%sql
/* Clean the database */

# drop the tables in a correct order to avoid foreign key constraints.
drop table if exists Checkpoint;
drop table if exists Submission;
drop table if exists IOPair;
drop table if exists Comment;
drop table if exists Post;
drop table if exists Problem;
drop table if exists User;
drop trigger if exists update_user_grade;
```

2. 建表

按照前文ER关系图建立所需的table，包含用户、问题、提交、IO答案、测试点、帖子、评论。

```
%%sql
/* Rebuid the database */

# The user table stores the information of users.
create table if not exists User (
    id          int primary key auto_increment,
    username    varchar(255) not null,
    password    varchar(255) not null,
    grade       enum('beginner', 'intermediate', 'advanced', 'expert') default 'beginner'
);

# Problems in the system.
create table if not exists Problem (
    id              int primary key auto_increment,
    title           varchar(255) not null,
    author_id       int,
    handin_time     timestamp default current_timestamp,
    description     text not null,
    difficulty      enum('easy', 'medium', 'hard') not null,
    time_limit      int not null default 1000, # ms
    memory_limit    int not null default 128, # MB
    # Do not delete the problem when the author is deleted.
    foreign key (author_id) references User (id) on delete set null
);

# A submission from a user to a problem.
create table if not exists Submission (
    id              int primary key auto_increment,
    problem_id      int not null,
    user_id         int,
    submit_time     timestamp default current_timestamp,
    code            text not null,
    language        enum('C/C++', 'Java', 'Python') not null,
    status          enum('passed', 'failed') default null,
    score           int default 0,
    foreign key (problem_id) references Problem (id) on delete cascade,
    foreign key (user_id) references User (id) on delete set null
);

# Input and output pairs of a problem.
create table if not exists IOPair (
    id          int not null, # weak entity
    problem_id  int not null,
    input       text not null,
    output      text not null,
    score       int not null,
    type        enum('sample', 'test') not null default 'test',
    # Notice: If the IO pair is a sample, the score does not matter.
    foreign key (problem_id) references Problem (id) on delete cascade,
    primary key (id, problem_id)  # composite primary key
);

# A checkpoint is a submission from user on specific IO pair.
create table if not exists Checkpoint (
    id              int primary key auto_increment,
    problem_id      int not null,
    iopair_id       int not null,
    submission_id   int not null,
    status          enum('P', 'AC', 'WA', 'TLE', 'MLE', 'RE', 'CE') not null default 'P',
    time_usage      int,
    memory_usage    int,
    info            text,
    score           int default 0,
    foreign key (iopair_id, problem_id) references IOPair (id, problem_id) on delete cascade,
    foreign key (submission_id) references Submission (id) on delete cascade
```

```
);

# A post is a message from a user commit under a problem.
create table if not exists Post
(
    id          int primary key auto_increment,
    title       varchar(255) not null,
    author_id   int,
    content     text not null,
    create_time timestamp default current_timestamp,
    problem_id  int,
    foreign key (author_id) references User (id) on delete set null,
    foreign key (problem_id) references Problem (id) on delete set null
);

# A comment is a message from a user commit under a post.
create table if not exists Comment
(
    id          int primary key auto_increment,
    post_id     int,
    author_id   int,
    content     text not null,
    create_time timestamp default current_timestamp,
    foreign key (post_id) references Post (id) on delete cascade,
    foreign key (author_id) references User (id) on delete set null
);
```

一个触发器，当用户通过足够多的题目时提升用户等级。

> 注：如遇报错，说明云端的服务器MySQL版本不能正常运行，经本机测试有效。
>
> 如需本地测试，在 配置JSON文件中修改数据库链接即可。

```
%%sql
create trigger if not exists update_user_grade
    after update
    on Submission
    for each row
begin
    declare passed_num int;
    select count(distinct problem_id)
    into passed_num
    from Submission
    where user_id = new.user_id
      and status = 'passed';
    if passed_num >= 4 then
        update User set grade = 'expert' where id = new.user_id;
    elseif passed_num >= 3 then
        update User set grade = 'advanced' where id = new.user_id;
    elseif passed_num >= 2 then
        update User set grade = 'intermediate' where id = new.user_id;
    end if;
end;
```

五、数据库服务

1. 插入数据

演示如何插入相关数据。
这里提供了一个简单的测试问题：计算两个整数之和，并提供了三个测试点。此外，还有用户发表的两个帖子和三个对帖子的

评论。

```
%%sql
/* Generate test data */

# User
insert into User (username, password)
values ('Admin', '123456');
insert into User (username, password)
values ('Alice', 'alice20050825');
insert into User (username, password)
values ('Bob', 'IDontCarePassword');

# Problem
insert into Problem (title, author_ID, description, difficulty, time_limit, memory_limit)
values ('A+B Problem', 1, '计算两整数之和。', 'easy', 1000, 256);
insert into IOPair (id, problem_id, input, output, score, type)
values (1, 1, '1 2', '3', 0, 'sample');
insert into IOPair (id, problem_id, input, output, score, type)
values (2, 1, '0 0', '0', 10, 'test');
insert into IOPair (id, problem_id, input, output, score, type)
values (3, 1, '-156 -1213', '-1369', 40, 'test');
insert into IOPair (id, problem_id, input, output, score, type)
values (4, 1, '6165481 84615613', '90781094', 50, 'test');

# Post
insert into Post (title, author_id, content, problem_id)
values ('A+B Problem的Python解答', 1, '很简单的问题，Python代码如下：
```python
a, b = map(int, input().split())
print(a + b)
```
', 1);
insert into Post (title, author_id, content, problem_id)
values ('Java方法', 1, '我用Java写的。Java的输入必须要新建一个Scanner对象
```java
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextint();
        int b = sc.nextInt();
        System.out.println(a + b);
    }
}
```
', 1);

Comment
insert into Comment (post_id, author_id, content)
values (1, 2, "tql! 第一次学会了用Python写程序，我之前写的都是C++不知道Python怎么转换类型");
insert into Comment (post_id, author_id, content)
values (1, 3, "我只是路过的，因为我还有数据库实习作业要写");
insert into Comment (post_id, author_id, content)
values (2, 3, "楼主有笔误啊，是`int a = sc.nextInt();`而不是`nextint();`");
```

```
%%sql
select * from User;
```

id	username	password	grade
1	Admin	123456	beginner

id	username	password	grade
2	Alice	alice20050825	beginner
3	Bob	IDontCarePassword	beginner

```
%%sql
select * from Problem;
```

id	title	author_id	handin_time	description	difficulty	time_limit	memory_limit
1	A+B Problem	1	2024-04-11 10:49:03	计算两整数之和。	easy	1000	256

```
%%sql
select * from IOPair;
```

id	problem_id	input	output	score	type
1	1	1 2	3	0	sample
2	1	0 0	0	10	test
3	1	-156 -1213	-1369	40	test
4	1	6165481 84615613	90781094	50	test

2. 对用户上传的代码进行评测

这是本次实习的核心函数。主要步骤：

- 从数据库中获取用户的提交内容
- 从数据库中获取问题的时间空间限制和测试用例
- 评测用户代码
- 把测评结果写回数据库

关于如何评测代码的细节，请参看 代码检查程序。

```python
from utils.code_checker import Code_Checker
from utils.data_helper import *


def submit_code(submission_id):
    """Let a user submit a code to a problem. The code will be tested by the checker."""
    # Get the submission info
    sql = f"select * from Submission where id = {submission_id};"
    cursor.execute(sql)
    submission = fetch_cursor(cursor)[0]
    problem_id, code, language = (
        submission["problem_id"],
        submission["code"],
        submission["language"],
    )

    # Get the problem info
    sql = f"select * from Problem where id = {problem_id};"
    cursor.execute(sql)
    problem = fetch_cursor(cursor)[0]
    time_limit, memory_limit = problem["time_limit"], problem["memory_limit"]

    # Get the IO pairs to test
    sql = f"select * from IOPair where problem_id = {problem_id};"
    cursor.execute(sql)
    io_pairs = fetch_cursor(cursor)

    question_status = "passed"
    question_score = 0
    for io_pair in io_pairs:
        # Get the IO info
        input_data, expected_output, score, io_type = (
            io_pair["input"],
            io_pair["output"],
            io_pair["score"],
            io_pair["type"],
        )
        if io_type == "sample":
            continue
        # Test the code
        checker = Code_Checker()
        checker.set_io(code, language, input_data, expected_output, submission_id)
        checker.set_limit(time_limit, memory_limit)
        status, info, used_time, used_memory = checker.test()
        if status != "AC":
            question_status = "failed"
            score = 0
        question_score += score
        # insert a new Checkpoint
        sql = (
            f"insert into Checkpoint (problem_id, iopair_id, submission_id, status, info, time_usage,
memory_usage, score) "
            f"values ({problem_id}, {io_pair['id']}, {submission_id}, '{status}', '{info}', {used_time},
{used_memory}, {score});"
        )
        cursor.execute(sql)

    # Update the submission
    sql = f"update Submission set status = '{question_status}', score = {question_score} where id =
{submission_id};"
    cursor.execute(sql)
    db.commit()
```

3. 测试效果

现在让用户上传一份代码

```
a,b=map(int,input().split())
print(a+b)
```

应当通过。另一份代码写成乘法了，应当产生结果错误，但依然能碰巧通过一个测试点。

```
%%sql
insert into Submission (problem_id, user_id, code, language)
values(1,2,"a,b=map(int,input().split())\nprint(a+b)",'Python'); # AC
insert into Submission (problem_id, user_id, code, language)
values(1,2,"a,b=map(int,input().split())\nprint(a*b)",'Python'); # WA
```

```
submit_code(1)
submit_code(2)
```

查看提交代码的信息和更详细的逐个测试点信息，正确代码得到满分，错误代码也能得到部分分数。

此外，如有需要，可以查看具体的测试点。

```
%%sql
select * from Submission;
```

id	problem_id	user_id	submit_time	code	language	status	score
1	1	2	2024-04-11 10:49:03	a,b=map(int,input().split()) print(a+b)	Python	passed	100
2	1	2	2024-04-11 10:49:03	a,b=map(int,input().split()) print(a*b)	Python	failed	10

```
%%sql
select * from Checkpoint;
```

id	problem_id	iopair_id	submission_id	status	time_usage	memory_usage	info	score
1	1	2	1	AC	135	4188	Answer Accepted	10
2	1	3	1	AC	127	4184	Answer Accepted	40
3	1	4	1	AC	116	4188	Answer Accepted	50
4	1	2	2	AC	134	4192	Answer Accepted	10
5	1	3	2	WA	121	4188	Line 1 Column 0: expected	0

id	problem_id	iopair_id	submission_id	status	time_usage	memory_usage	info	score
							-..., but got 1.	
6	1	4	2	WA	129	4188	Line 1 Column 0: expected 9..., but got 5.	0

4. 更多数据

现在进行一些数据库的查询，为避免大段代码插入数据，选择从CSV读取，然后将其插入到数据库中。

> 注：评测十几份代码可能会花费一二十秒的时间。
>
> 如遇报错，大概率是命令行指令错误。请确保电脑中含有g++, python3, java编译器。

```
setURL(config["user"], config["password"], config["host"], config["port"], config["db"])
csv2sql("./data/user.csv", "user")  # see the function in data_helper.py
csv2sql("./data/problem.csv", "problem")
csv2sql("./data/post.csv", "post")
csv2sql("./data/comment.csv", "comment")
csv2sql_IOpair("./data/iopair.csv", "iopair")
csv2sql_submission("./data/submission.csv", "submission")

# prepare for the submission
sql = "select count(*) from Submission;"
cursor.execute(sql)
submission_num = cursor.fetchone()[0]

# NOTE: MAY COST A LONG TIME HERE!
# Make sure you have python interpreter, javac and g++ compiler on your PC
for i in range(3, submission_num + 1):
    print("testing submission", i, "...")
    submit_code(i)
print("All submissions have been tested!")
```

```
%%sql
# avoid a too long code from being shown in the chart
select problem_id,user_id,submit_time,CONCAT(LEFT(code,20), "...") as code,language,status,score from Submission
limit 7;
```

problem_id	user_id	submit_time	code	language	status	score
1	2	2024-04-11 10:49:03	a,b=map(int,input()....	Python	passed	100
1	2	2024-04-11 10:49:03	a,b=map(int,input()....	Python	failed	10
1	4	2018-03-09 00:00:00	print("Hello, World!...	Python	failed	0
2	2	2018-03-09 00:00:00	#include <iostream> ...	C/C++	passed	100
2	3	2018-03-09 00:00:00	k = int(input())	Python	passed	100

problem_id	user_id	submit_time	code	language	status	score
			ans...			
2	3	2018-03-09 00:00:00	import java.util.Sca...	Java	failed	0
2	4	2018-03-09 00:00:00	#include <iostream> ...	C/C++	failed	0

5. 触发器测试

查看所有用户，如果SQL版本适配，按照触发器，Alice应成为intermediate，Bob应成为advanced（原CSV文件中没有指定用户等级，默认都是beginner）。

> 注：云端的SQL版本不适配可能会失效。

```
%%sql
select * from User;
```

id	username	password	grade
1	Admin	123456	beginner
2	Alice	alice20050825	intermediate
3	Bob	IDontCarePassword	advanced
4	Benjamin Ward	SantaClaus	beginner
5	Yang120	19891220abc	beginner
6	Nobody	qwerty	beginner
7	LaoDaXiangNiLe	KobeHelicopter	beginner

六、数据库查询

- 查询问题的所有提交

```
%%sql
set @problem_id = 1; # The problem id you want to query.
select * from Submission where problem_id = @problem_id;
```

id	problem_id	user_id	submit_time	code	language	status	score
1	1	2	2024-04-11 10:49:03	a,b=map(int,input().split()) print(a+b)	Python	passed	100
2	1	2	2024-04-11 10:49:03	a,b=map(int,input().split()) print(a*b)	Python	failed	10
3	1	4	2018-03-09 00:00:00	print("Hello, World!")	Python	failed	0

- 查询用户所有的通过题目的提交

```
%%sql

select user_id, problem_id, id as submission_id
from Submission
where Submission.status = 'passed'
```

user_id	problem_id	submission_id
2	1	1
2	2	4
3	2	5
3	3	11
4	5	14
3	5	15

- 查询最热门的题目

```
%%sql
select problem_id as hot_problem_id
from Submission S
group by problem_id
order by count(*) desc
limit 1;
```

hot_problem_id
2

- 查询通过率（AC数/提交数）最高的用户

```
%%sql
select User.*
from User
join Submission on User.id = Submission.user_id
group by User.id
order by count(
    case
        when Submission.status = 'passed' then 1
    end
) * 1.0 / count(*) desc
limit 1;
```

id	username	password	grade
3	Bob	IDontCarePassword	advanced

- 按照评论数排序某一道题下的热门评论，这里以第三题为例

```
%%sql

set @problem_id = 3; # The problem id you want to query.
select Post.id, Post.content, comment_count
from Post join (select post_id, count(*) as comment_count
                from Comment
                group by post_id) C on Post.id = C.post_id
where Post.problem_id = @problem_id
union all
select Post.id, Post.content, 0 as comment_count
from Post
where Post.problem_id = @problem_id
  and Post.id not in (select post_id
                      from Comment);
```

id	content	comment_count
5	这一题将A-B=C转换成A-C=B，首先将A数组每个元素出现的次数统计起来，用map映射， 最后将A数组每次减一个C，再将A数组扫一遍，将所有映射的次数和加起来就是答案	1
6	双指针，一个指向A数组，一个指向B数组，如果A-B<C，那么A指针右移，如果A-B>C， 那么B指针右移，如果A-B=C，那么A指针右移，B指针右移，直到A或B指针到达数组末尾	3
7	楼下的都是用的二分查找，我用二分查找做了一次。 但我的A-B数对的代码也能AC， 没有TLE，挺黑科技的。map求解。将每一个数字映射到map中， 答案每次加上num[i]+c位置的数的个数，num[i]+c即为式子的另一个加数	0

- 按评论数量对用户进行排序

```
%%sql
select author_id as most_comment_user, count(*) as comment_count
from comment
group by author_id
order by count(*) desc
```

most_comment_user	comment_count
2	5
3	4
4	2
1	1
5	1