|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 《数据库原理》实验报告 | | | | |
| 题目：实验八  数据库综合实验 | 学号：2020302978  2020302977 | 姓名：  罗权  张建龙 | 班级：  10012003  10012007 | 日期：12/17 |

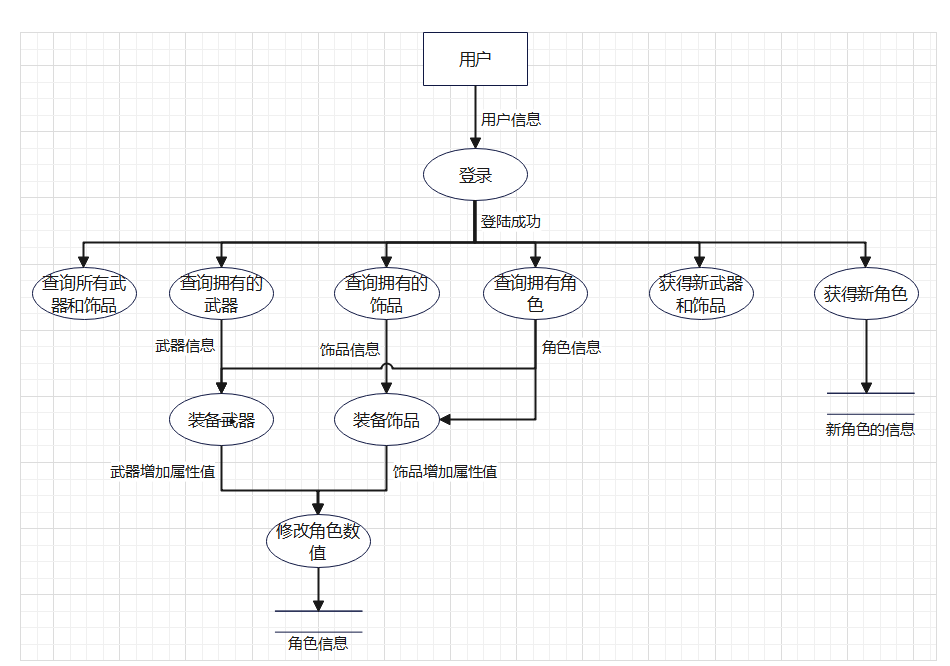
1. 实验内容、步骤以及结果

## 实验目的

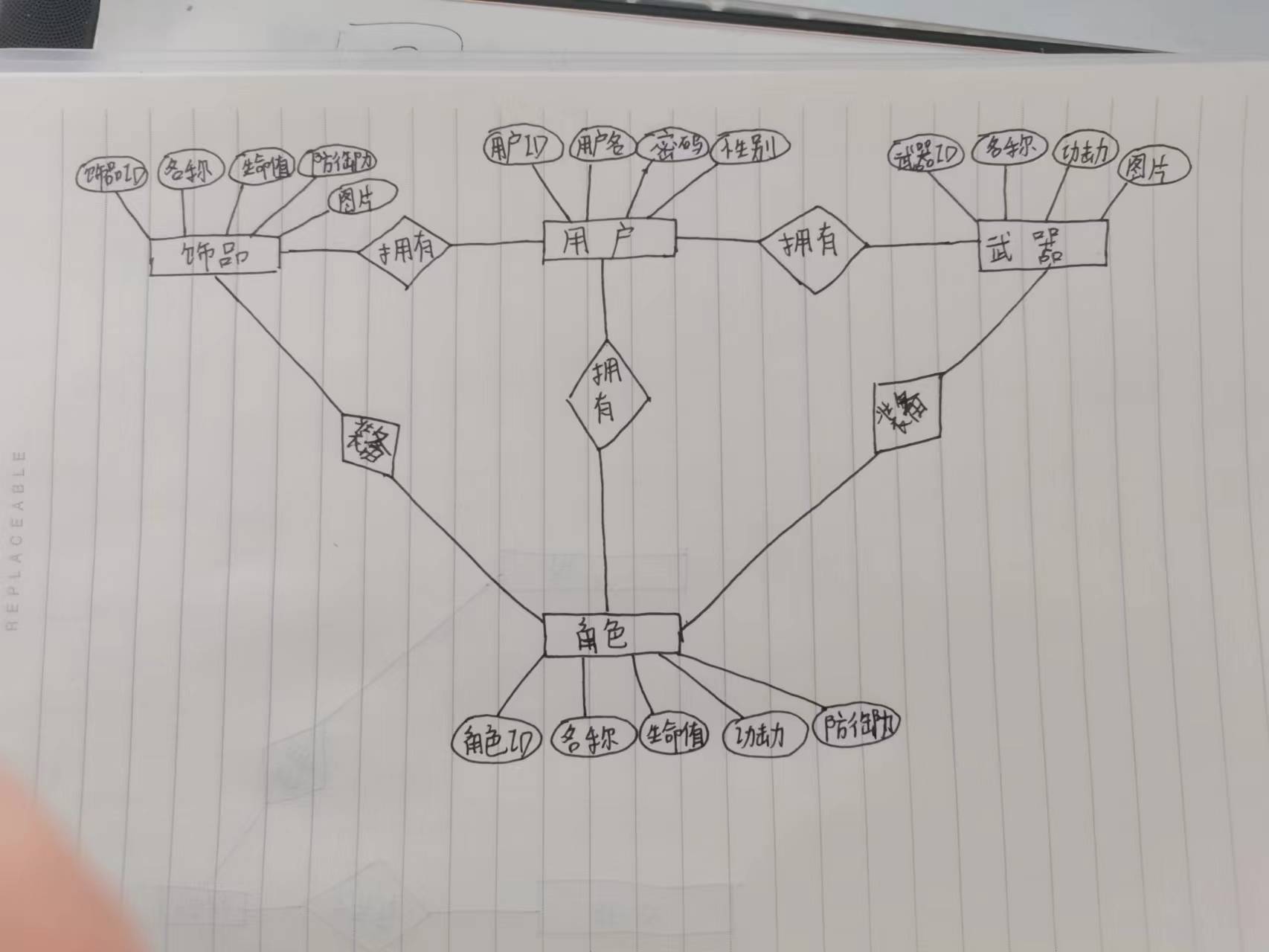
通过完成从用户需求分析、概念结构设计、逻辑结构设计等一系列的数据库设计到编程、调试和应用等全过程，进一步理解和掌握教材中的相关内容。

## 实验要求

1. **实验步骤**
2. 需求分析：实现一个可以给用户展示角色以及角色的属性，如攻击力、防御力和生命力，同时用于还可以查看自己拥有的武器和饰品，以及它们所带来的增益效果。管理员可以进行用户的一切操作，除了给角色穿戴武器和饰品。管理员还可以增加/修改角色、武器及饰品的信息。
3. 数据流图和数据字典：



1. E-R图：



1. 关系模式：

用户（用户ID，用户名，登录密码，用户性别）；

角色（角色ID，角色名，生命值，攻击力，防御力）；

饰品（饰品ID，饰品名，生命值，防御力，饰品图片）；

武器（武器ID，武器名，攻击力，武器图片）；

用户\_角色（用户ID，角色ID，武器ID，饰品ID）；

用户\_饰品（用户ID，饰品ID，角色ID）；

用户\_武器（用户ID，武器ID，角色ID）；

1. 主键约束：用户的主键是用户ID，角色的主键是角色ID，饰品的主键是饰品ID，武器的主键是武器ID，用户\_角色的主键是(用户ID，角色ID)，用户\_饰品的主键是(用户ID，饰品ID)，用户\_武器的主键是(用户ID，武器ID)

外键约束：用户\_角色表的外键是用户ID，角色ID，武器ID和饰品ID；用户\_饰品表的外键是用户ID和饰品ID；用户\_武器表的外键是用户ID和武器ID。这些外键约束是用来防止在绑定用户、角色、饰品和武器之间的对应关系的时候出现错误。

检查约束：为用户表的性别属性增加一个Check约束，用来约束性别只能为男或者为女。

索引：用户\_角色表建立关于(角色ID，武器ID，饰品ID)的索引，这样可以加速用户查询角色穿戴的武器以及饰品的信息；用户\_武器表建立关于(角色ID，武器ID)的索引，可以加速用户查找武器的穿戴状态以及被谁穿戴的过程；用户\_饰品表建立关于(角色ID，饰品ID)，可以加速用户查找饰品穿戴状态以及被谁穿戴的过程。

触发器：删除武器的触发器dele\_weapon，这个触发器在管理员删除武器的时候，会同时删除掉用户\_角色以及用户\_武器表中的关于此武器的数据；删除饰品的触发器dele\_accessories，这个触发器则是在管理员删除饰品的时候，会同时删除掉用户\_角色以及用户\_饰品表中的关于此饰品的数据。

其余的操作都是依靠java完成。

语言：java SE 8

平台：windows 10

数据库：MySQL 8.0.22、

二．数据库端的部分代码解释如下：

在管理员端，我为几个用途相似的功能设计了存储过程：

CREATE PROCEDURE update\_WA(IN name\_project VARCHAR(10),IN id VARCHAR(10),IN name\_new VARCHAR(10),IN hp\_new int,IN attack\_new int,IN defense\_new int,IN img VARCHAR(100),IN flag tinyint)

BEGIN

-- 修改武器的信息

if flag = 0 THEN

UPDATE weapon set wId = id,wname = name\_new,attack = attack\_new,wimg = img WHERE wname = name\_project;

ELSE IF flag = 1 THEN

UPDATE accessories set aId = id,aname = name\_new,hp = hp\_new,defense = defense\_new,aimg = img WHERE aname = name\_project;

END if;

END//

这个存储过程是管理员在更新武器或者饰品的信息的时候，使用的一个存储过程。在更新武器或者饰品的信息的时候，我们需要提前设置一个flag ，然后存储过程会根据调用的时候，传入的flag的值来进行物品信息的更新。

在用户端，我也设计了几个存储过程：

DROP PROCEDURE if EXISTS Get\_New;

delimiter //

CREATE PROCEDURE Get\_New(IN uId\_new int,IN wId\_new int,IN cId\_new int,IN aId\_new int,IN flag TINYINT)

BEGIN

-- 添加角色

if flag = 0 THEN

INSERT INTO user\_own\_charactor (uId,cId,aId,wId) SELECT uId\_new,cId\_new,NULL,NULL WHERE not EXISTS(SELECT uId,cId FROM user\_own\_charactor WHERE uId = uId\_new AND cId = cId\_new);

-- 添加饰品

ELSEIF flag = 1 THEN

INSERT INTO user\_own\_accessories (uId,aId,cId) SELECT uId\_new,aId\_new,NULL WHERE not EXISTS(SELECT uId,aId FROM user\_own\_accessories WHERE uId = uId\_new AND aId = aId\_new);

ELSEIF flag = 2 THEN

INSERT INTO user\_own\_weapon (uId,wId,cId) SELECT uId\_new,wId\_new,NULL WHERE not EXISTS(SELECT uId,wId FROM user\_own\_weapon WHERE uId = uId\_new AND wId = wId\_new);

END IF;

END//

delimiter ;

这个存储过程是在用户获得新的角色、武器或者饰品的时候，被调用的一个存储过程。同时，我还在这里面添加了一个判断，只有当在我们要插入的表中找不到我们要插入的数据时，负责这个数据是不能够添加进数据库的。

-- 饰品

delimiter //

CREATE PROCEDURE Wear\_Accessories(IN u\_Id VARCHAR(20),IN a\_ID VARCHAR(10),IN c\_ID VARCHAR(10))

BEGIN

DECLARE cId\_temp VARCHAR(10);

DECLARE hp\_temp int;

DECLARE defense\_temp int;

-- 根据这个查询的值来判断这个饰品是否有角色带

SELECT cId INTO cId\_temp FROM user\_own\_accessories WHERE uId = u\_Id and aId = a\_ID;

-- 没有人带

if cId\_temp is null THEN

BEGIN

SELECT hp into hp\_temp FROM accessories WHERE aId = a\_ID;

SELECT defense into defense\_temp FROM accessories WHERE aId = a\_ID;

UPDATE user\_own\_accessories SET cId = c\_ID WHERE uId = u\_Id AND aId = a\_ID;

UPDATE user\_own\_charactor SET aId = a\_ID WHERE uId = u\_Id AND cId = c\_ID;

-- 下面的语句可以改为触发器来执行（修改角色的生命和防御）

UPDATE charactor SET hp = (hp + hp\_temp),defense = (defense + defense\_temp) WHERE cid = c\_ID;

END;

ELSEIF cId\_temp is not null THEN

BEGIN

SIGNAL SQLSTATE '02000'

set MESSAGE\_TEXT = '该饰品已经被佩戴了';

END;

END if;

END //

delimiter ;

这个存储过程是在为角色穿戴是饰品的时候调用的，我们首先要判断即将要被穿戴的饰品是否已经被该用户拥有的其他角色穿戴，如果已经被穿戴，则报错“不能够穿戴”；否则直接更新用户\_角色表的饰品ID字段以及用户\_饰品表的角色ID字段即可。

delimiter //

CREATE PROCEDURE Wear\_Weapon(IN u\_Id VARCHAR(20),IN w\_ID VARCHAR(10),IN c\_ID VARCHAR(10))

BEGIN

DECLARE cId\_temp VARCHAR(10);

DECLARE attack\_temp int;

-- 根据这个查询的值来判断这个武器是否有角色带

SELECT cId INTO cId\_temp FROM user\_own\_weapon WHERE uId = u\_Id and wId = w\_ID;

-- 没有人带

if cId\_temp is null THEN

BEGIN

SELECT attack into attack\_temp FROM weapon WHERE wId = w\_ID;

UPDATE user\_own\_weapon SET cId = c\_ID WHERE uId = u\_Id AND wId = w\_ID;

UPDATE user\_own\_charactor SET wId = w\_ID WHERE uId = u\_Id AND cId = c\_ID;

-- 下面的语句可以改为触发器来执行（修改角色的攻击力）

UPDATE charactor SET attack = (attack + attack\_temp) WHERE cid = c\_ID;

END;

ELSEIF cId\_temp is not null THEN

BEGIN

SIGNAL SQLSTATE '02000'

set MESSAGE\_TEXT = '该武器已经被佩戴了';

END;

END if;

END //

delimiter ;

这个存储过程是在为角色穿戴是武器的时候调用的，我们首先要判断即将要被穿戴的武器是否已经被该用户拥有的其他角色穿戴，如果已经被穿戴，则报错“不能够穿戴”；否则直接更新用户\_角色表的武器ID字段以及用户\_武器表的角色ID字段即可。

1. 实验中出现的问题以及解决方案(对于未解决问题请将问题列出来)

除了标题内容以外，该部分内容中还可以写对于实验的一些感受，建议，意见等。

第一个问题：ID的字符串不能使用字符串，要使用int类型，因为只有int类型能够实现自增的功能。在添加新的角色的时候，id属性是不能让我们自己填的，因为若数据过多，那么我们可能导致id重复，从而无法插入数据。

第二个问题：在插入数据的时候，我们首先要判断这个数据在表中是否存在，如存在则不插入，不存在则插入。这里有两种方法，第一种是先select这个数据，然后使用if语句进行判断来决定是否插入；第二种直接使用INSERT语句，但是要在后面加上select 要插入的数据 where not exists (select 要插入的数据 from 表名 where 判断条件) 来进行判断是否插入，但是在第一次写的时候，在查找资料的时候查找出错，在select 要插入的数据 后面加上了一个from 表名 导致始终插入不了数据，最终在查看了多篇博客之后，找到了解决方法。

**批阅者：**

**批阅日期：**

**实验成绩：**

**批注：**