数据库Lab1

PB20000126 葛哲凯

一、创建上述基本表, 并插入部分测试数据

创建四个表成功：

运行insert_data.sql，插入数据成功:

二、用 SQL 语言完成下面小题,并测试运行结果

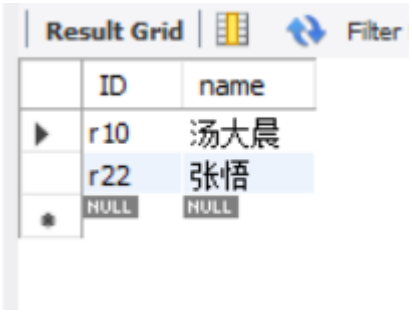
（1） 查询读者 Rose 借过的读书（包括已还和未还）的图书号、书名和借期

```
SELECT b.ID, b.name, br.borrow_Date
FROM Book b, Borrow br, Reader r
WHERE b.ID = br.book_ID AND br.reader_ID = r.ID AND r.name = 'Rose';
```

ID	name	borrow_Date
b1	数据库系统实现	2022-02-22
b11	三体	2022-01-11
b16	中国2185	2022-01-11
b19	HowWeThink	2023-04-08
b2	数据库系统概念	2022-02-22

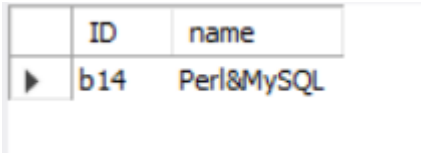
（2） 查询从没有借过图书也从没有预约过图书的读者号和读者姓名

```
SELECT r.ID, r.name
FROM Reader r
WHERE r.ID NOT IN (
  SELECT DISTINCT reader_ID
  FROM Borrow
  UNION
  SELECT DISTINCT reader_ID
  FROM Reserve
);
```



(3) 查询被借阅次数最多的作者（注意一个作者可能写了多本书）

```
SELECT b.author, SUM(b.borrow_Times) AS total_borrow_times
FROM Book b
GROUP BY b.author
ORDER BY total_borrow_times DESC
LIMIT 1;
```



(4) 查询目前借阅未还的书名中包含“MySQL”的的图书号和书名

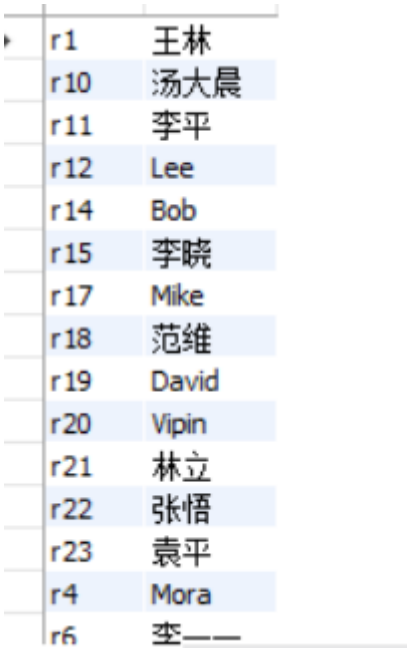
```
SELECT b.ID, b.name
FROM Book b, Borrow br
WHERE b.ID = br.book_ID AND b.name LIKE '%MySQL%'AND br.return_Date IS NULL;
```

(5) 查询借阅图书数目超过 10 本的读者姓名

```sql
SELECT reader.name
from reader
where reader.ID in(
select reader.ID
FROM reader, borrow
WHERE reader.ID = borrow.reader_ID
GROUP BY reader.ID
HAVING count(*) > 10);
```

(6) 查询没有借阅过任何一本 John 所著的图书的读者号和姓名

```sql
SELECT r.ID, r.name
FROM Reader r
WHERE NOT EXISTS (
    SELECT *
    FROM Book b
    WHERE b.author = 'John'
    AND b.ID IN (
        SELECT book_ID
        FROM Borrow br
        WHERE br.reader_ID = r.ID
    )
);
```



r1	王林
r10	汤大晨
r11	李平
r12	Lee
r14	Bob
r15	李晓
r17	Mike
r18	范维
r19	David
r20	Vipin
r21	林立
r22	张悟
r23	袁平
r4	Mora
r6	李——

（7） 查询 2022 年借阅图书数目排名前 10 名的读者号、姓名以及借阅图书；

```sql
SELECT r.ID, r.name, COUNT(*) AS borrow_count
FROM Reader r, Borrow b
WHERE r.ID = b.reader_ID
AND YEAR(b.borrow_Date) = 2022
GROUP BY r.ID
ORDER BY borrow_count
LIMIT 10;
```

ID	name	borrow_count
r11	李平	4
r2	Rose	4
r3	罗永平	4
r1	王林	3
r23	袁平	3
r7	王二狗	3
r8	赵四	3
r9	魏心	3
h5	汤晨	2
r13	Jack	2

（8） 创建一个读者借书信息的视图,该视图包含读者号、姓名、所借图书号、图 书名和借期；并使用该视图查询最近一年所有读者的读者号以及所借阅的 不同图书数

```sql
CREATE VIEW ReaderBorrowInfo AS
SELECT r.ID AS reader_ID, r.name AS reader_name, b.book_ID, bk.name AS book_name,
b.borrow_Date
FROM Reader r, Borrow b, Book bk
WHERE r.ID = b.reader_ID AND b.book_ID = bk.ID;

# 并使用该视图查询最近一年所有读者的读者号以及所借阅的不同图书数
SELECT reader_ID, COUNT(DISTINCT book_ID) AS book_count
FROM ReaderBorrowInfo
WHERE YEAR(borrow_Date) = YEAR(NOW())-1
GROUP BY reader_ID;
```

h5	3
r11	4
r12	1
r13	2
r14	2
r15	1
r16	1
r17	1
r19	1
r2	1
r23	3
r4	1
r6	2
r9	2

三、设计一个存储过程 updateReaderID,实现对读者表的 ID 的修改

- 首先向READER表中插入一条新记录（new_id,name,age,address)以避免外键冲突
- 再更新BORROW表和RESERVE表中reader_id = old_id相关的记录
- 最后删去READER表中ID = old_id的记录，更新完成
- 为避免报错：You are using safe update mode and you tried to update a tablewithout a WHERE that uses a KEY column To disable safe mode, toggle the optionin Preferences -> SQL Editor and reconnect.需要先将安全模式关闭，最后再开启

```sql
DELIMITER //
CREATE PROCEDURE updateReaderID(
  IN old_id CHAR(8),
  IN new_id CHAR(8)
)
BEGIN
DECLARE tempname varchar(10);
DECLARE tempage int;
DECLARE tempaddress varchar(20);
```

```
SET SQL_SAFE_UPDATES=0;
start transaction;
    select name,age,address from READER where ID = old_id into
tempname,tempage,tempaddress;
    INSERT INTO Reader(ID,name,age,address)
Values(new_id,tempname,tempage,tempaddress);

    UPDATE borrow SET reader_ID = new_id WHERE reader_ID = old_id;
    UPDATE reserve SET reader_ID = new_id WHERE reader_ID = old_id;

    Delete From Reader where ID = old_id;
    COMMIT;
    SET SQL_SAFE_UPDATES=0;
END //
DELIMITER ;
```

四、设计一个存储过程 borrowBook, 当读者借书时调用该存储过程完成借书处理

- 设置s作为检测是否有冲突的哨兵，每当检测到非法情况发生，则在s后面沾上报错字符串
- 依次检测实验文档中abc三种情况，并设置报错输出
- d情况删除操作，即使在表中没有找到对应的记录，删除操作也不会报错
- 使用START TRANSACTION --- ROLLBACK 结构，使得有非法情况出现时可以撤销操作
- 如果出现主键冲突，则需要使用系统定义异常SQLEXECEPTION

```
drop procedure if exists borrowBook;
DELIMITER //
CREATE PROCEDURE borrowBook(IN bookID CHAR(8), IN readerID CHAR(8), IN borrowDate
DATE, OUT state CHAR(100))
BEGIN
    DECLARE borrowCount INT;
    DECLARE reserveCount INT;
    DECLARE i INT DEFAULT 1;
    DECLARE text varchar(100);
    DECLARE bookstatus INT;
    Declare s varchar(10) default '';
    declare continue HANDLER FOR SQLEXCEPTION SET s = '4';
    START TRANSACTION;
    -- 执行DML语句
    -- 检查同一天不允许同一个读者重复借阅同一本读书
    SELECT COUNT(*) INTO borrowCount FROM borrow WHERE reader_ID = readerID AND
book_ID = bookID AND borrow_Date = borrowDate;
    IF borrowCount > 0 THEN
        SET s = IFNULL(concat(s, 'a'),'');
    END IF;

    -- 如果该图书存在预约记录，而当前借阅者没有预约，则不允许借阅
```

```
    -- 先检测当前reader是否有预约记录，如果没有，检查book是否被别人预约
    SELECT COUNT(*) INTO reserveCount FROM reserve WHERE book_ID = bookID AND
reader_ID = readerID;
     IF reserveCount = 0 THEN
        SELECT COUNT(*) INTO reserveCount FROM reserve WHERE book_ID = bookID AND
reader_ID != readerID;
            IF reserveCount > 0 THEN
            set s = IFNULL(concat(s , 'b'),'');
        END IF;
    END IF;

    -- 一个读者最多只能借阅 3 本图书
    SELECT COUNT(*) INTO borrowCount FROM borrow WHERE reader_ID = readerID AND
return_Date IS NULL;
    IF borrowCount >= 3 THEN
       set s = IFNULL(concat(s, 'c'),'');
    END IF;

    -- 如果图书的status是2，即已经被借阅，则不能借阅
    SELECT status into bookstatus from book where ID = bookID;
    IF bookstatus = 1 THEN
        set s = IFNULL(concat(s, 'd'),'');
    END IF;

    -- 如果借阅者已经预约了该图书，则删除预约记录
    delete from reserve WHERE book_ID = bookID AND reader_ID = readerID;

    -- 借阅成功后修改图书表中的 status
    UPDATE Book SET status = 1, borrow_Times = borrow_Times + 1 WHERE ID = bookID;

    -- 插入借阅记录updateReaderIDupdateReaderID
    INSERT INTO Borrow (book_ID, reader_ID, borrow_Date) VALUES (bookID, readerID,
borrowDate);

    IF s = '' THEN
        set state = 'Success!';
        COMMIT;
    ELSE
        SET text = '';
        while MID(s,i,1) != '' DO
            CASE MID(s,i,1)
                WHEN 'a' THEN SET text = concat(text,'同一天不允许同一个读者重复借阅同一本
读书! \n');
                WHEN 'b' THEN SET text = concat(text,'如果该图书存在预约记录，而当前借阅者
没有预约，则不允许借阅! \n');
                WHEN 'c' THEN SET text = concat(text,'一个读者最多只能借阅 3 本图书，意味
着如果读者已经借阅了 3 本图书并
且未归还则不允许再借书! \n');
                WHEN 'd' THEN SET text = concat(text,'图书正在被借阅中! ');
                ELSE SET text = 'SQLEXEPTION';
            END CASE;
            SET i = i + 1;
```

```
        END WHILE;
        SET state = text;
        ROLLBACK;
    END IF;
END //
DELIMITER ;
```

五、设计一个存储过程 returnBook，当读者还书时调用该存储过程完成还书处理。

思路与四相似

```
DelIMITER //
CREATE PROCEDURE returnBook(IN readerID CHAR(8), IN bookID CHAR(8), IN returnDate
DATE, OUT state CHAR(100))
BEGIN
    DECLARE returnCount INT;
    DECLARE borrowCount INT;
    DECLARE bookstatus INT;
    DECLARE i INT DEFAULT 1;
    DECLARE text varchar(100);
    Declare s varchar(10) default '';
    declare continue HANDLER FOR SQLEXCEPTION SET s = '4';

     START TRANSACTION;
    -- 执行DML语句

    -- 同一天不允许同一个读者重复返还同一本读书
    SELECT COUNT(*) INTO returnCount FROM borrow WHERE reader_ID = readerID AND
book_ID = bookID AND return_Date = returnDate;
    IF returnCount > 0 THEN
        SET s = IFNULL(concat(s, 'a'),'');
    END IF;

    -- 如果该图书存在借阅记录，而且并没有被返还，而且借阅者不是当前读者，则不允许返还
    SELECT COUNT(*) INTO borrowCount FROM borrow WHERE book_ID = bookID AND
reader_ID != readerID AND return_Date = NULL;
    IF borrowCount > 0 THEN
        set s = IFNULL(concat(s , 'b'),'');
    END IF;

    -- 如果图书的status不是1，即图书不处于被借阅的状态，则不能被返还
    select status into bookstatus from book where ID = bookID;
    IF bookstatus != 1 THEN
        SET s = IFNULL(concat(s,'c'),'');
    END IF;
```

```
    -- 如果借阅者已经借阅了该图书，则允许返还，但要求返还完成后设置借阅表中的 return_Date 为返还
日期
    update borrow SET return_Date = returnDate where reader_id = readerID AND
book_id = bookID;

    -- 返还成功后修改 status
    update book SET status = 0 where ID = bookID;

    IF s = '' THEN
        set state = 'Success!';
        COMMIT;
    ELSE
        SET text = '';
        while MID(s,i,1) != '' DO
            case MID(s,i,1)
                WHEN 'a' THEN SET text = concat(text,'同一天不允许同一个读者重复返还同一本
读书! \n');
                WHEN 'b' THEN SET text = concat(text,'如果该图书存在借阅记录，而且并没有被
返还，而且借阅者不是当前读者，则不允许返还! \n');
                WHEN 'c' THEN SET text = concat(text,'图书未被借阅中! ');
                ELSE SET text = 'SQLEXEPTION';
            END CASE;
            SET i = i + 1;
        END WHILE;
        SET state = text;
        ROLLBACK;
    END IF;
END //
DELIMITER ;
```

6、 设计一个触发器

```
DELIMITER //
CREATE TRIGGER book_reserve_trigger
AFTER INSERT ON reserve
FOR EACH ROW
BEGIN
    -- 预约时将图书状态改为 2，增加预约次数
    UPDATE Book SET status = 2, reserve_Times = reserve_Times + 1 WHERE ID =
NEW.book_ID;
END;
//
CREATE TRIGGER book_unreserve_trigger
AFTER DELETE ON reserve
FOR EACH ROW
BEGIN
    -- 取消预约时减少预约次数
    DECLARE Times INT;
    UPDATE book SET reserve_Times = reserve_Times - 1 WHERE ID = OLD.book_ID;
    -- 当预约次数为0时，书的status转为0
```

```
    select reverse_Times into Times FROM book where ID = OLD.book_ID;
    IF Times = 0 THEN
        update book SET status = 0;
    END IF;
END;
//
CREATE TRIGGER book_borrow_trigger
AFTER INSERT ON Borrow
FOR EACH ROW
BEGIN
    -- 借出时减少预约次数
    UPDATE book SET reserve_Times = 0, status = 1 WHERE ID = NEW.book_ID;
END;
//
```

生成triggers前，需要先登录root用户，执行：

```
set global log_bin_trust_function_creators = 1;
```

完成后，再set = 0.否则运行triggers.sql时会报错You do not have the SUPER privilege and binary logging is enabled