数据库系统及应用实验 1 报告

傅申 PB20000051

1. 实验目的

在 MySQL 上创建一个图书馆数据库.

2. 实验环境

OS Manjaro Linux 6.1.26-1

Database MySQL Version 8.0.33 for Linux on x86_64 (Source distribution)

IDE JetBrains DataGrip 2023.1.1

3. 实验过程

3.1. 创建基本表

```sql
create database if not exists lab1;
use lab1;

create table if not exists Book (
    ID char(8) primary key,
    name varchar(10) not null,
    author varchar(10),
    price float,
    status int check (status in (0, 1, 2)),
    borrow_Times int default 0,
    reserve_Times int default 0
);

create table if not exists Reader (
    ID char(8) primary key,
    name varchar(10),
    age int,
    address varchar(20)
);

create table if not exists Borrow (
    book_ID char(8) references Book(ID),
    reader_ID char(8) references Reader(ID),
    borrow_Date date,
    return_Date date,
    primary key (book_ID, reader_ID, borrow_Date)
);

create table if not exists Reserve (
    book_ID char(8),
    reader_ID char(8),
    reserve_Date date,
    take_Date date,
    primary key (book_ID, reader_ID, reserve_Date),
    check (take_Date ⩾ reserve_Date)
);
```

3.2. 用 SQL 语言完成小题

(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 Book.ID, Book.name, Borrow.borrow_date
from Book join Borrow on Book.ID = Borrow.book_ID
          join Reader on Borrow.reader_ID = Reader.ID
where Reader.name = 'Rose';
```

(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 ID, name
from Reader
where ID not in (select distinct reader_ID from Borrow) and
      ID not in (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 author
from Book
group by author
order by sum(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 book_ID, name
from Book join Borrow on Book.ID = Borrow.book_ID
where Borrow.return_Date is null and
      Book.name like '%MySQL%';
```

(5) 查询借阅图书数目超过 10 本的读者姓名;

```
select name
from Reader join Borrow on 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;

```
select ID, name
from Reader
where ID not in (
    select distinct reader_ID
    from Borrow join Book on Borrow.book_ID = Book.ID
```

```sql
    where Book.author = 'John'
    );
```

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

```sql
select reader_ID, name, count(*) as book_num
from Reader join Borrow on Reader.ID = Borrow.reader_ID
where year(Borrow.borrow_date) = 2022
group by reader_ID
order by book_num desc
limit 10;
```

(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;

```sql
create or replace view reader_book as
select reader_ID, Reader.name as reader_name, book_ID,
       Book.name as book_name, borrow_date
from Borrow join Reader on Borrow.reader_ID = Reader.ID
            join Book on Borrow.book_ID = Book.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.

```sql
select reader_ID, count(distinct book_ID) as book_num
from reader_book
where date(borrow_date) between date_sub(curdate(), interval 1 year)
                                and curdate()
group by reader_ID;
```

3.3. 设计存储过程 `updateReaderID`

在尝试更新读者号时, 可能会出现如下的情况:

- 成功
- 失败
  - 旧的读者号不存在
  - 新的读者号已经存在
- SQL 内部错误
  - Warning
  - Exception

在更新过程中, 如果旧的读者号存在, 并且新的读者号不存在, 则要在 `Reader` 表中更新对应表项的 `ID`, 并且还要更新 `Borrow` 和 `Reserve` 中的 `reader_ID`. 过程将对各种情况进行处理, 只有旧的读者号存在, 并且新的读者号不存在时才 `commit`, 否则 `rollback`, 过程会使用输出 `state` 和用户变量 `@info` 指示更新的结果:

```sql
create procedure updateReaderID(in old_id char(8), in new_id char(8),
                                out state int)
    modifies sql data
```

```
begin
    declare old_id_exists, new_id_exists bool default false;

    # s:
    #   0: able to update
    #   1: old_id not found
    #   2: new_id exists
    #   3: SQL warning
    #   4: SQL exception
    declare s int default 0;
    declare continue handler for sqlwarning set s = 3;
    declare continue handler for sqlexception set s = 4;

    start transaction;
    select exists(select * from Reader where ID = old_id) into old_id_exists;
    select exists(select * from Reader where ID = new_id) into new_id_exists;

    if s = 0 then
        if not old_id_exists then
            set s = 1;
        elseif new_id_exists then
            set s = 2;
        else
            update Reader set ID = new_id where ID = old_id;
            update Borrow set reader_ID = new_id where reader_ID = old_id;
            update Reserve set reader_ID = new_id where reader_ID = old_id;
        end if;
    end if;

    if s = 0 then
        set state = 0;
        set @info = 'Update success';
        commit;
    else
        case s
            when 1 then set state = 1;
                        set @info = concat('Reader ', old_id, ' not found');
            when 2 then set state = 2;
                        set @info = concat('Reader ', new_id,
                                           ' already exists');
            when 3 then set state = 3;
                        set @info = 'SQL warning';
            when 4 then set state = 4;
                        set @info = 'SQL exception';
            end case;
        rollback;
    end if;
end;
```

3.4. 设计存储过程 `borrowBook`

在尝试借书时, 可能会出现如下的情况:

- 借书成功
  - 读者没有对应书的预约记录
  - 读者有对应书的预约记录
- 借书失败
  - 读者不存在
  - 书不存在
  - 书已借出
  - 读者今天已经借了该书
  - 读者没有该书的预约记录, 但是该书被预约
  - 读者已经借了 3 本书
- SQL 内部错误
  - Warning
  - Exception

在借书过程中, 如果读者被允许借书, 则要在 `Borrow` 表中插入新的表项, 并且修改 `Book` 表中对应书的 `status` 和 `borrow_Times`, 如果有预约, 则还要删除 `Reserve` 中对应的表项. 过程将对各种情况进行处理, 只有读者被允许借书才 `commit`, 否则 `rollback`. 过程会使用输出 `state` 和用户变量 `@info` 指示借书的结果:

```sql
drop procedure if exists borrowBook;
create procedure borrowBook(in rid char(8), in bid char(8), out state int)
    modifies sql data
begin
    declare reader_exists, book_exists bool default false;
    declare book_status int default 0;
    declare reader_last_borrow_date date default null;
    declare book_reserved_by_reader bool default false;
    declare books_reader_borrowed int default 0;

    # s:
    #   0: the reader is able to borrow the book without reservation
    #   1: the reader is able to borrow the book with reservation
    #   2: reader not found
    #   3: book not found
    #   4: the book is not available
    #   5: the reader has already borrowed the book today
    #   6: the reader has no reservation for the book but the book is reserved
    #   7: the reader has already borrowed 3 books
    #   8: SQL warning
    #   9: SQL exception
    declare s int default 0;

    declare continue handler for sqlwarning set s = 8;
    declare continue handler for sqlexception set s = 9;

    start transaction;

    select exists(select * from Reader where ID = rid) into reader_exists;
    select exists(select * from Book where ID = bid) into book_exists;

    if not reader_exists then
        set s = 2;
    elseif not book_exists then
        set s = 3;
    else
        select status from Book where ID = bid into book_status;

        select max(borrow_date)
        from Borrow
        where reader_ID = rid and book_ID = bid
        into reader_last_borrow_date;

        select exists(
            select *
            from Reserve
            where reader_ID = rid and book_ID = bid
        ) into book_reserved_by_reader;

        select count(*)
        from Borrow
        where reader_ID = rid and return_date is null
        into books_reader_borrowed;

        if reader_last_borrow_date = curdate() then
            set s = 5;
        elseif books_reader_borrowed ≥ 3 then
            set s = 7;
        elseif book_status = 0 then # not reserved or borrowed
            set s = 0;
        elseif book_status = 1 then # borrowed
            set s = 4;
        elseif book_status = 2 then # reserved
            if book_reserved_by_reader then
                set s = 1;
            else
                set s = 6;
            end if;
        end if;
    end if;

    if s < 2 then
        insert into Borrow values (bid, rid, curdate(), null);
        update Book set status = 1 where ID = bid;
        update Book set borrow_Times = borrow_Times + 1 where ID = bid;
        if s = 1 then
            delete from Reserve where book_ID = bid and reader_ID = rid;
        end if;
    end if;

    if s < 2 then
```

```
        case s
            when 0 then set state = 0;
                        set @info = concat('Book ', bid,
                                           ' borrowed by reader ', rid,
                                           ' without reservation.');
            when 1 then set state = 1;
                        set @info = concat('Book ', bid,
                                           ' borrowed by reader ', rid,
                                           ' with reservation.');
            end case;
        commit;
    else
        case s
            when 2 then set state = 2;
                        set @info = concat('Reader ', rid, ' not found.');
            when 3 then set state = 3;
                        set @info = concat('Book ', bid, ' not found.');
            when 4 then set state = 4;
                        set @info = concat('Book ', bid,
                                           ' is not available now.');
            when 5 then set state = 5;
                        set @info = concat('Reader ', rid,
                                           ' has already borrowed book ', bid,
                                           ' today.');
            when 6 then set state = 6;
                        set @info = concat('Reader ', rid,
                                           ' has no reservation for book ',
                                           bid, ' but the book is reserved.');
            when 7 then set state = 7;
                        set @info = concat('Reader ', rid,
                                           ' has already borrowed 3 books.');
            when 8 then set state = 8;
                        set @info = concat('SQL warning.');
            when 9 then set state = 9;
                        set @info = concat('SQL exception.');
            end case;
        rollback;
    end if;
end;
```

3.5. 设计存储过程 `returnBook`

在尝试还书时, 可能会出现如下的情况:

- 还书成功
- 还书失败
  - 读者不存在
  - 书不存在
  - 读者并没有借这本书
- SQL 内部错误
  - Warning

- Exception

在还书过程中, 如果读者被允许还书, 则要在 `Borrow` 表中更新对应表项的 `return_Date`, 并修改 `Book` 表中对应表项的 `status`. 过程将对各种情况进行处理, 只有读者被允许还书才 `commit`, 否则 `rollback`. 过程会使用输出 `state` 和用户变量 `@info` 指示还书的结果:

```sql
create procedure returnBook(in rid char(8), in bid char(8), out state int)
    modifies sql data
begin
    declare reader_exists, book_exists, borrow_exists bool default false;
    declare book_reserved bool default false;
    declare new_status int default 0;

    # s:
    #     0: able to return
    #     1: reader not found
    #     2: book not found
    #     3: the reader has not borrowed this book
    #     4: SQL warning
    #     5: SQL exception
    declare s int default 0;
    declare continue handler for sqlwarning set s = 4;
    declare continue handler for sqlexception set s = 5;

    start transaction;

    select exists(select * from Reader where ID = rid) into reader_exists;
    select exists(select * from Book where ID = bid) into book_exists;

    if not reader_exists then
        set s = 1;
    elseif not book_exists then
        set s = 2;
    else
        select exists(select *
                      from Borrow
                      where reader_ID = rid
                        and book_ID = bid
                        and return_Date is null)
        into borrow_exists;
        if borrow_exists then
            select exists(select *
                          from Reserve
                          where reader_ID = rid
                            and book_ID = bid)
            into book_reserved;
            if book_reserved then
                set new_status = 2;
            else
                set new_status = 0;
            end if;
        else
```

```sql
                set s = 3;
            end if;
        end if;

        if s = 0 then
            update Borrow
            set return_Date = curdate()
            where reader_ID = rid
              and book_ID = bid
              and return_Date is null;
            update Book
            set status = new_status
            where ID = bid;
        end if;

        if s = 0 then
            set state = 0;
            set @info = concat('Reader ', rid, ' returned book ', bid, '.');
            commit;
        else
            case s
                when 1 then set state = 1;
                            set @info = concat('Reader ', rid, ' not found.');
                when 2 then set state = 2;
                            set @info = concat('Book ', bid, ' not found.');
                when 3 then set state = 3;
                            set @info = concat('Reader ', rid,
                                               ' has not borrowed Book ', bid,
                                               '.');
                when 4 then set state = 4;
                            set @info = 'SQL warning.';
                when 5 then set state = 5;
                            set @info = 'SQL exception.';
            end case;
            rollback;
        end if;
    end;
```

3.6. 预约触发器

使用两个触发器 `new_reservation` 和 `cancel_reservation` 来实现, 其中

- `new_reservation` 在 `Reserve` 被插入表项时触发, 在书的 `status` 不为 `1` 时更新 `status` 为 `2`, 并给 `reserve_times` 加 1.
- `cancel_reservation` 在 `Reserve` 被删除表项时触发, 在书的 `status` 不为 `1` 且 `reserve_times` 为 `1` 时更新 `status` 为 `0`, 并给 `reserve_times` 减 1.

```sql
create trigger new_reservation after insert on Reserve for each row
begin
    declare old_status int default 0;
    declare old_reserve_times int default 0;

    select status
    from Book
    where ID = new.book_ID
    into old_status;

    select reserve_Times
    from Book
    where ID = new.book_ID
    into old_reserve_times;

    if old_status ≠ 1 then
        update Book
        set status = 2
        where ID = new.book_ID;
    end if;

    update Book
    set reserve_Times = old_reserve_times + 1
    where ID = new.book_ID;
end;

create trigger cancel_reservation after delete on Reserve for each row
begin
    declare old_reserve_times int default 0;
    declare old_status, new_status int default 0;

    select reserve_Times
    from Book
    where ID = old.book_ID
    into old_reserve_times;

    select status
    from Book
    where ID = old.book_ID
    into old_status;

    if old_status = 1 then
        set new_status = 1;
    elseif old_reserve_times ⩽ 1 then
        set new_status = 0;
    else
        set new_status = 2;
    end if;

    update Book
    set status = new_status, reserve_Times = old_reserve_times - 1
    where ID = old.book_ID;
end;
```

4. 实验结果

4.1. 创建基本表并插入测试数据

运行正常, 运行后创建了 `Book`, `Reader`, `Borrow`, `Reserve` 四个表, 并分别插入了 19, 23, 85, 3 条数据.

4.2. 用 SQL 语言完成小题

(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;

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;

ID	name
r10	汤大晨
r22	张悟

(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);

author
刘慈欣

(4) 查询目前借阅未还的书名中包含"MySQL"的的图书号和书名;

book_ID	name
b14	Perl&MySQL

(5) 查询借阅图书数目超过 10 本的读者姓名;

name
王林
王林
David

(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;

ID	name
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	李一一
r8	赵四

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

reader_ID	**name**	**book_num**
r11	李平	4
r2	Rose	4
r3	罗永平	4
r1	王林	3
r7	王二狗	3
r9	魏心	3
r8	赵四	3
r23	袁平	3
r4	Mora	2
r6	李一一	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.

reader_ID	**book_num**
r11	4
r12	1
r13	2
r14	2
r15	1
r16	1
r17	1
r19	1

r2	1
r23	3
r4	1
r5	3
r6	2
r9	2

4.3. 存储过程 `updateReaderID`

- 更新失败的情况
  - 旧的读者号不存在

    比如 `call updateReaderID('r0', 'r25', @state);`, 输出的 `@state` 和 `@info` 分别为 `1` 和 `Reader r0 not found`
  - 新的读者号已经存在

    比如 `call updateReaderID('r1', 'r2', @state);`, 输出的 `@state` 和 `@info` 分别为 `2` 和 `Reader r2 already exists`
- 更新成功的情况 比如 `call updateReaderID('r1', 'r24', @state);`, 输出的 `@state` 和 `@info` 分别为 `0` 和 `Update success`, 并且 `Reader`, `Borrow` 表中都有相应的修改.

4.4. 存储过程 `borrowBook`

- 借书失败的情况
  - 在没有预约的情况下借被预约的书

    比如 `call borrowBook('r1', 'b10', @state);`, 输出的 `@state` 和 `@info` 分别为 `6` 和 `Reader r1 has no reservation for book b10 but the book is reserved.`
  - 借了 3 本书的读者尝试借书

    比如 `call borrowBook('r23', 'b7', @state);`, 输出的 `@state` 和 `@info` 分别为 `7` 和 `Reader r23 has already borrowed 3 books.`
  - 尝试借一本已经被借出的书

    比如 `call borrowBook('r1', 'b1', @state);`, 输出的 `@state` 和 `@info` 分别为 `4` 和 `Book b1 is not available now.`
- 借书成功的情况
  - 未预约

    比如 `call borrowBook('r1', 'b7', @state);`, 输出的 `@state` 和 `@info` 分别为 `0` 和 `Book b7 borrowed by reader r1 without reservation.`, 并且在 `Borrow` 表中插入了相应的记录, `Book` 表中 `b7` 对应的表项中 `status` 和 `borrow_Times` 也被修改为 `0` 和 `5`.
  - 已预约

4.5. 存储过程 `returnBook`

- 还书失败的情况
  - 读者并没有借书

    比如 `call returnBook('r1', 'b1', @state);`, 输出的 `@state` 和 `@info` 分别为 `3` 和 `Reader r1 has not borrowed Book b1.`

- 还书成功的情况
  比如 `call returnBook('r14', 'b1', @state);`, 输出的 `@state` 和 `@info` 分别为 `0` 和 `Reader r14 returned book b1.`, 且 `Borrow` 中对应的表项的 `return_Date` 被修改为今天, `Book` 中 `b1` 对应表项的 `status` 被修改为 `0`.

4.6. 触发器

4.6.1. `new_reservation`

书 `b12` 在 `Book` 中的表项为

ID	name	author	price	status	borrow_Times	reserve_Times
b12	Fun python	Luciano	354.2	0	3	0

在向 `Reserve` 插入 4 条预约信息后

```sql
insert into Reserve values('b12', 'r20', curdate() - 1, null);
insert into Reserve values('b12', 'r21', curdate() - 1, null);
insert into Reserve values('b12', 'r20', curdate(), null);
insert into Reserve values('b12', 'r21', curdate(), null);
```

表项变为了

ID	name	author	price	status	borrow_Times	reserve_Times
b12	Fun python	Luciano	354.2	2	3	4

与预期一致.

4.6.2. `cancel_reservation`

接着上面的操作, 删除 3 条预约信息后

```sql
delete from Reserve where book_ID = 'b12' and reserve_date = curdate();
delete from Reserve where book_ID = 'b12' and reader_ID = 'r21';
```

表项变为了

ID	name	author	price	status	borrow_Times	reserve_Times
b12	Fun python	Luciano	354.2	2	3	1

再让 `r20` 借出 `b12` 后

```sql
call borrowBook('r20', 'b12', @state);
```

表项变为了

ID	name	author	price	status	borrow_Times	reserve_Times
b12	Fun python	Luciano	354.2	1	3	0

与预期一致.

5. 总结与思考

本报告给出了一个图书馆管理系统的数据库设计, 并给出了相应的实现.