数据库 Lab1 实验报告

PB20111686 黄瑞轩

1 创建表

1.1 书籍表

```sql
CREATE TABLE Book(
    -- 主键是 ID
    ID CHAR(8) NOT NULL PRIMARY KEY,
    -- 姓名不能为空
    name VARCHAR(10) NOT NULL,
    author VARCHAR(10),
    price FLOAT,
    -- status、次数的默认值为0
    status INT DEFAULT 0,
    borrow_Times INT DEFAULT 0,
    reserve_Times INT DEFAULT 0,
    -- status 的取值只能是 0, 1, 2
    CONSTRAINT status CHECK(status >= 0 AND status <= 2)
);
```

创建结果:

Book

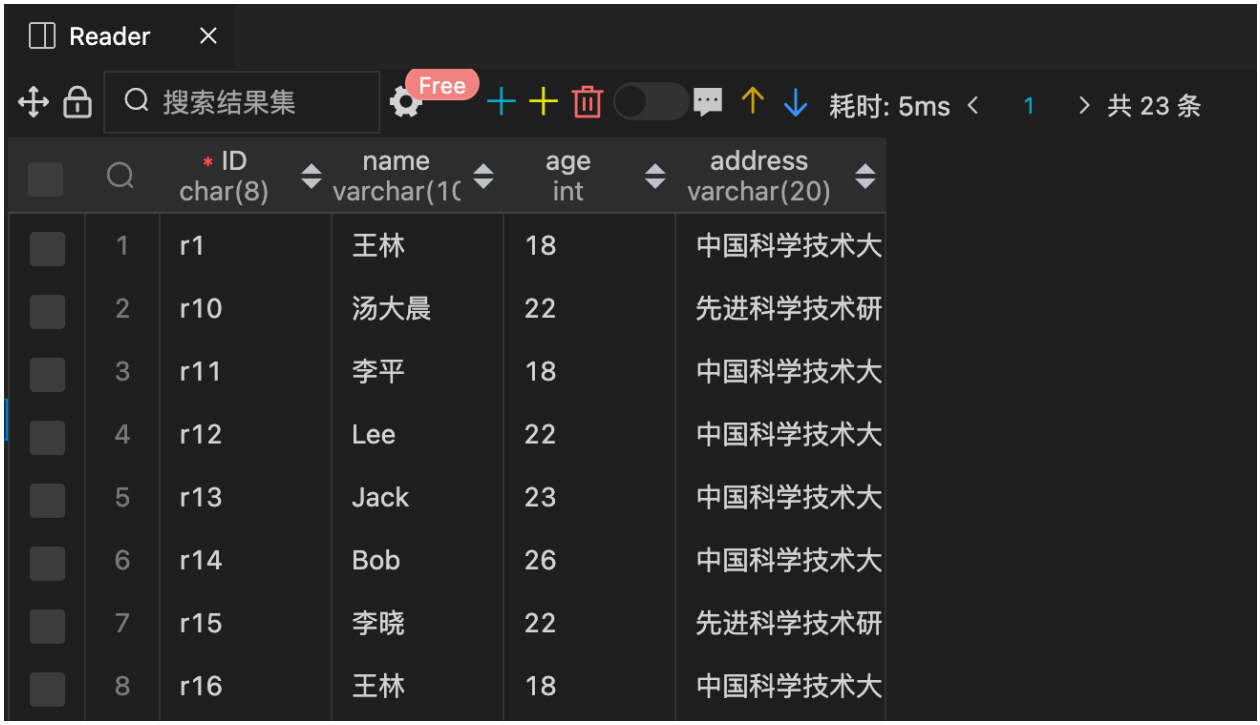
搜索结果集 Free 耗时: 5ms 1 共 19 条

	* ID char(8)	* name varchar(10)	author varchar(1(	price float	status int	borrow_Times int	reserve_Times int
1	b1	数据库系统实	Ullman	59	1	8	0
2	b10	数理逻辑	汪芳庭	22	2	6	4
3	b11	三体	刘慈欣	23	2	8	2
4	b12	Fun python	Luciano	354.2	0	3	0
5	b13	Learn SQL	Seyed	23	1	3	0
6	b14	Perl&MySQL	徐泽平	23	1	3	0
7	b15	司马迁的故事	黄永年	34.8	0	2	0
8	b16	中国2185	刘慈欣	218.5	1	7	0

1.2 读者表

```sql
CREATE TABLE Reader(
    -- 主键是 ID
    ID CHAR(8) NOT NULL PRIMARY KEY,
    name VARCHAR(10),
    age INT,
    address VARCHAR(20)
);
```

创建结果:



	* ID char(8)	name varchar(1(	age int	address varchar(20)
1	r1	王林	18	中国科学技术大
2	r10	汤大晨	22	先进科学技术研
3	r11	李平	18	中国科学技术大
4	r12	Lee	22	中国科学技术大
5	r13	Jack	23	中国科学技术大
6	r14	Bob	26	中国科学技术大
7	r15	李晓	22	先进科学技术研
8	r16	王林	18	中国科学技术大

1.3 借书表

```sql
CREATE TABLE Borrow(
    book_ID CHAR(8),
    reader_ID CHAR(8),
    borrow_Date DATE,
    return_date DATE,
    -- 主键
    CONSTRAINT PRIMARY KEY (book_ID, reader_ID, borrow_Date),
    -- 外键
    CONSTRAINT FK_BKID FOREIGN KEY (book_ID) REFERENCES Book(ID),
    CONSTRAINT FK_RDID FOREIGN KEY (reader_ID) REFERENCES Reader(ID)
);
```

Borrow

搜索结果集 Free 耗时: 5ms 1 共 85 条

	book_ID char(8)	reader_ID char(8)	borrow_Date date	return_date date
1	b1	r1	2022-03-12	2022-04-07
2	b1	r11	2022-07-01	2022-07-05
3	b1	r14	2023-01-10	(NULL)
4	b1	r16	2019-09-11	2019-11-09
5	b1	r2	2022-02-22	2022-03-10
6	b1	r3	2022-04-14	2022-06-19
7	b1	r7	2022-01-01	2022-01-20
8	b1	r9	2022-06-20	2022-06-30

1.4 预约表

```sql
CREATE TABLE Reserve(
    book_ID CHAR(8),
    reader_ID CHAR(8),
    -- 将预约日期默认设置为当前日期
    reserve_Date DATE DEFAULT (CURRENT_DATE),
    take_Date DATE,
    -- 主键
    CONSTRAINT PRIMARY KEY (
        book_ID,
        reader_ID,
        reserve_Date
    ),
    -- 外键
    CONSTRAINT FK_BKID2 FOREIGN KEY (book_ID) REFERENCES Book(ID),
    CONSTRAINT FK_RDID2 FOREIGN KEY (reader_ID) REFERENCES Reader(ID),
    -- 检查：预约取书日期不能晚于预约日期
    CONSTRAINT CHK_TKDATE CHECK (take_Date >= reserve_Date)
);
```

Reserve

搜索结果集 Free 耗时: 7ms 1 共 3 条

	book_ID char(8)	reader_ID char(8)	reserve_Date date	take_Date date
1	b10	r20	2023-02-17	(NULL)
2	b10	r21	2023-02-18	(NULL)
3	b11	r20	2023-02-17	(NULL)

2 表查询

2.1 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

Rose 的借书记录在 Borrow 表中，Borrow 表中只有读者号、图书号，需要使用连接查询。

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

查询结果:

Result

搜索结果集 Free 耗时: 2ms 1 共 5 条

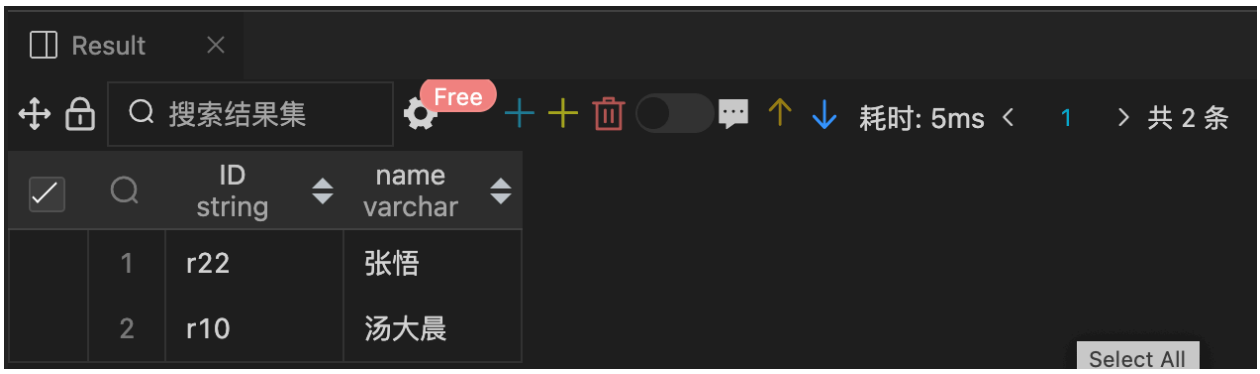
	ID string	name varchar	borrow_Date date
1	b1	数据库系统实	2022-02-22
2	b11	三体	2022-01-11
3	b16	中国2185	2022-01-11
4	b19	HowWeThink	2023-04-08
5	b2	数据库系统概	2022-02-22

2.2 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

使用 NOT IN 语句可以从 Reader 表中排除借过或预约过的读者信息。

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

查询结果:

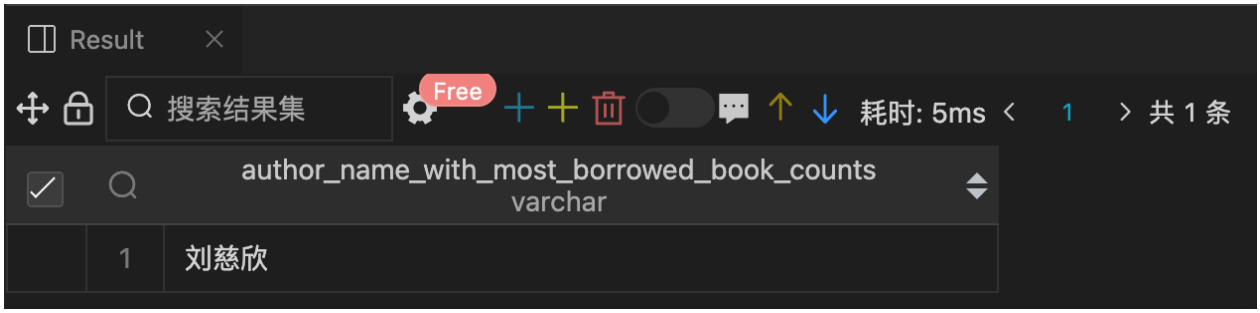


2.3 查询被借阅次数最多的作者(注意一个作者可能写了多本书)

首先，在子查询中计算每个作者的借阅总数，然后从中可选出借阅次数最大值；在父查询中查找借阅总数等于此最大值的作者名字即可。

```sql
SELECT author AS author_name_with_most_borrowed_book_counts
FROM Book
GROUP BY author
HAVING SUM(borrow_Times) = (
        SELECT
            MAX(total_borrow_Times)
        FROM (
                SELECT
                    SUM(borrow_Times) AS total_borrow_Times
                FROM Book
                GROUP BY
                    author
            ) AS Total
    );
```

查询结果:

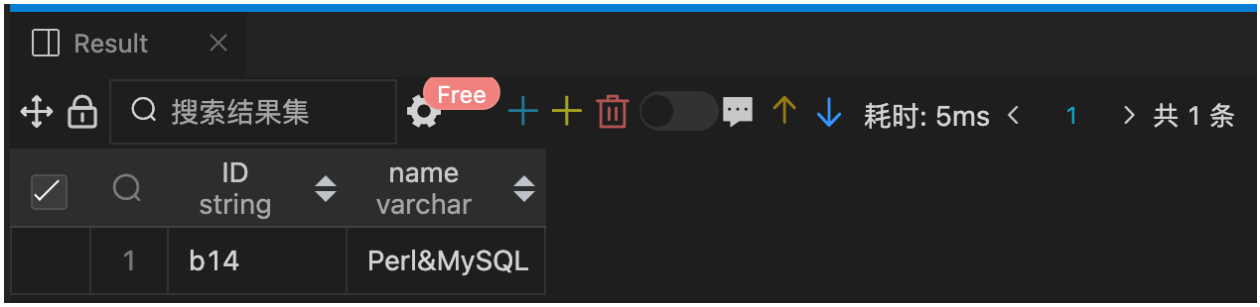


2.4 查询目前借阅未还的书名中包含“MySQL”的的图书号和书名

「借阅未还」指的是 return_date 为 NULL 的借阅记录，书名包含 MySQL 可用 LIKE 语句表示。

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

查询结果:

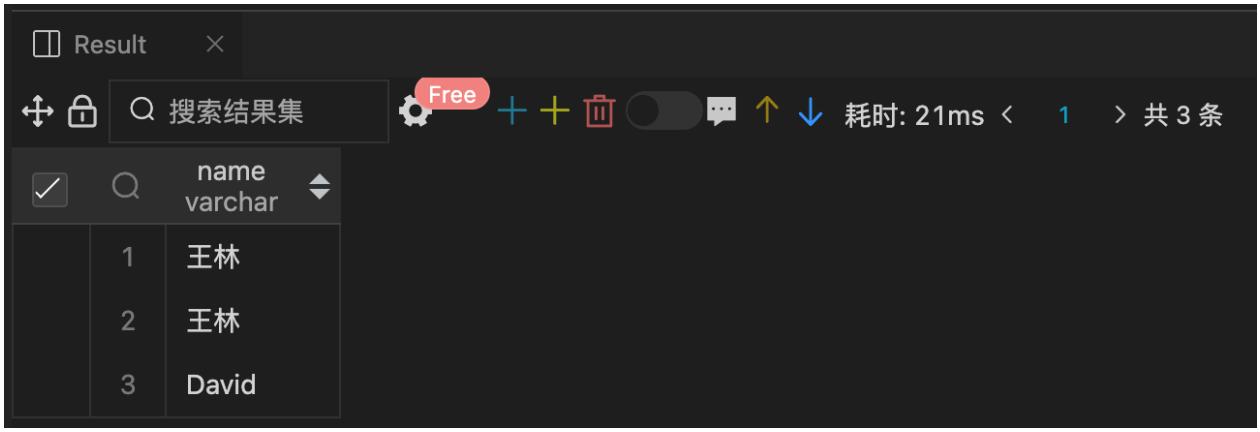


2.5 查询借阅图书数目超过 10 本的读者姓名

在通过读者号分组求和，并且用 HAVING 子句表达数目超过 10。

```
SELECT Reader.name AS name
FROM Borrow, Reader
WHERE
    YEAR(Borrow.borrow_Date) = 2022
    AND Borrow.reader_ID = Reader.ID
GROUP BY Borrow.reader_ID
HAVING COUNT(*) > 10;
```

查询结果: （有重名的）

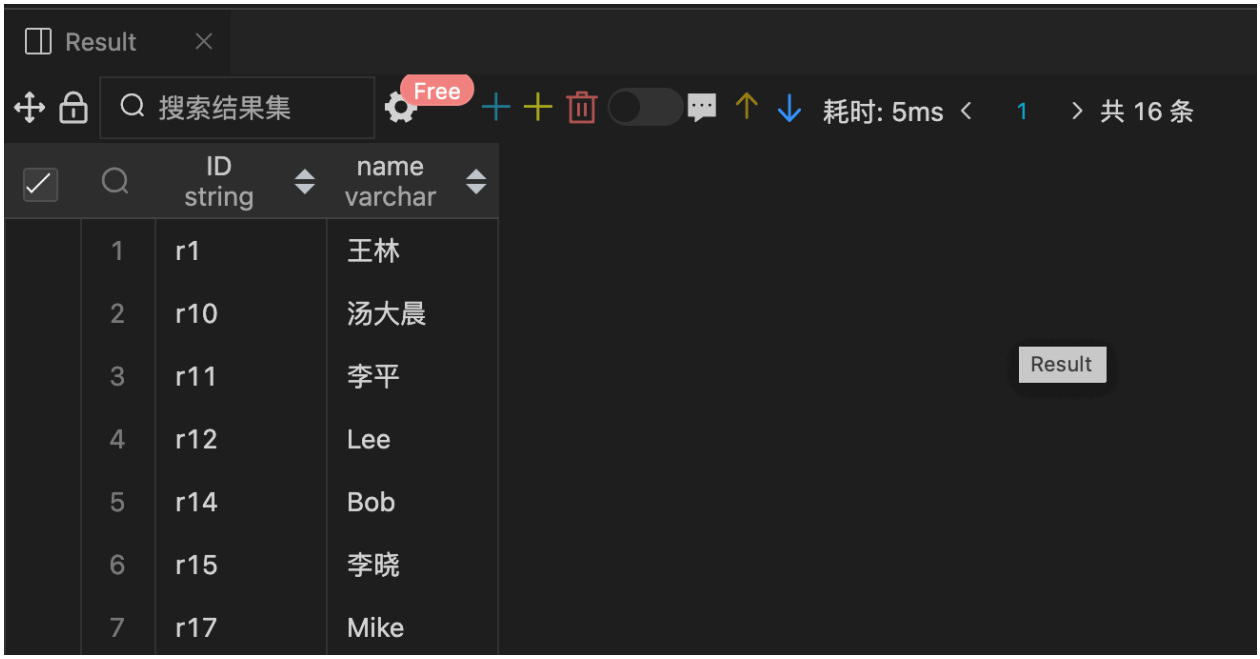


2.6 查询没有借阅过任何一本 John 所著的图书的读者号和姓名

使用 NOT EXISTS 从 Reader 表（不能从 Borrow 表，因为有的读者可能从未借过书）中排除。

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

查询结果:



2.7 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

使用 LIMIT 10 输出前 10 位查询信息，使用 desc 指定降序排序，使用 BETWEEN ... AND ... 确定借书日期在 2022 年。

```sql
SELECT ID, name, COUNT(bkID) AS borrow_times
FROM (
        SELECT
            Reader.ID AS ID,
            Reader.name AS name,
            Borrow.book_ID AS bkID
        FROM Reader, Borrow
        WHERE
            Borrow.reader_ID = Reader.ID
            AND borrow_Date BETWEEN '2022-01-01' AND '2022-12-31'
    ) AS Total
GROUP BY ID, name
ORDER BY borrow_times DESC
LIMIT 10;
```

查询结果:

Result

搜索结果集 Free 耗时: 3ms 1 2 共 17 条

	ID string	name varchar	borrow_times bigint
1	r11	李平	4
2	r2	Rose	4
3	r3	罗永平	4
4	r1	王林	3
5	r8	赵四	3
6	r7	王二狗	3
7	r9	魏心	3
8	r23	袁平	3

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

```sql
CREATE VIEW BORROW_INFOS AS
    SELECT
        Reader.ID AS reader_ID,
        Reader.name AS reader_name,
        Book.ID AS book_ID,
        Book.name AS book_name,
        Borrow.borrow_Date AS borrow_date
    FROM Reader, Book, Borrow
    WHERE
        Reader.ID = Borrow.reader_ID
        AND Book.ID = Borrow.book_ID;
```

创建结果:

borrow_infos

搜索结果集 Free 耗时: 8ms 1 共 85 条

	reader_II char(8)	reader_name varchar(10)	book_ID char(8)	book_na varchar(1(	borrow_date date
1	r1	王林	b1	数据库系统实	2022-03-12
2	r11	李平	b1	数据库系统实	2022-07-01
3	r14	Bob	b1	数据库系统实	2023-01-10
4	r16	王林	b1	数据库系统实	2019-09-11
5	r2	Rose	b1	数据库系统实	2022-02-22

- 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

  使用 DATE_SUB 函数可以将当前日期直接减去 1 年，而不用管具体细节。

```sql
SELECT
    reader_ID,
    COUNT(DISTINCT book_ID) AS borrow_book_cnt
FROM borrow_infos
WHERE
    borrow_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY reader_ID;
```

查询结果:

borrow_infos

搜索结果集 Free 耗时: 6ms 1 共 14 条

	reader_II char(8)	borrow_book_cnt bigint
1	r11	4
2	r12	1
3	r13	2
4	r14	2
5	r15	1

3 存储过程

因为这些过程都涉及多个表或者多个元组的修改，所以需要使用事务进行编程。

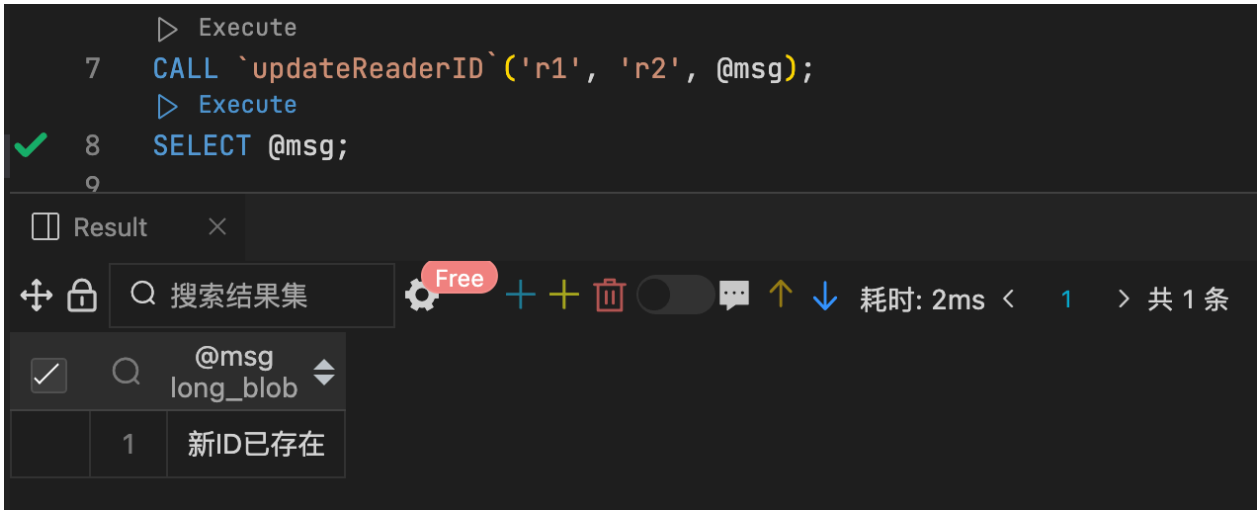
3.1 设计一个存储过程 updateReaderID 实现对读者表的 ID 的修改

由于外键约束不能直接修改，做法是先获得有旧 ID 的 Reader 元组，再将选出的元组中旧 ID 改为新 ID，再将修改过的元组插入 Reader 表。此时可以将 Borrow、Reserve 中的读者 ID 更新，更新之后将 Reader 表中的旧元组删除。

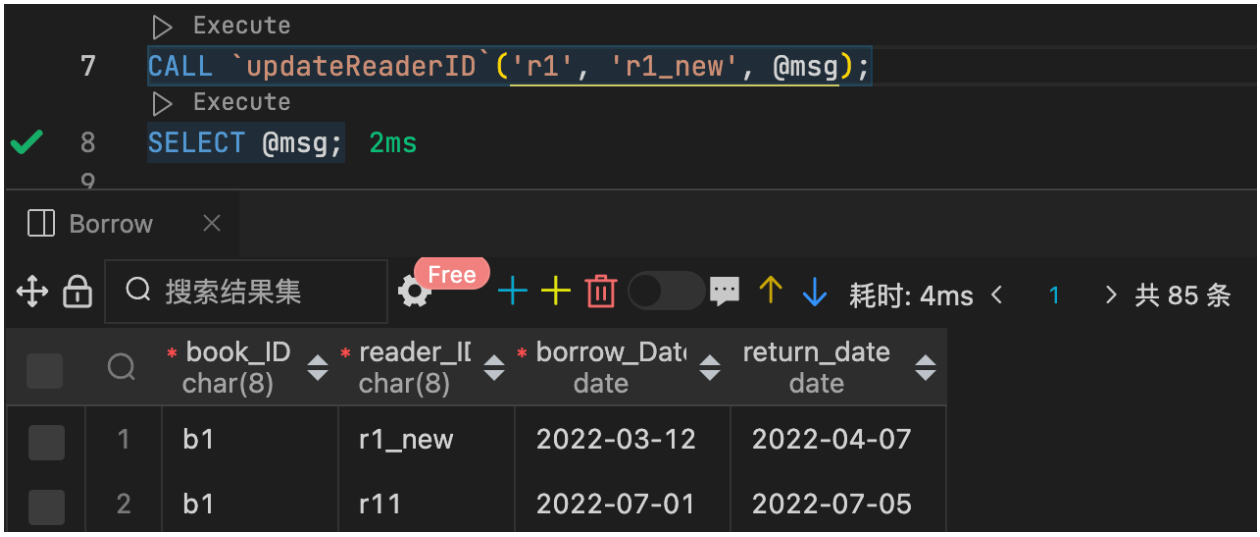
```sql
CREATE PROCEDURE updateReaderID(
    IN old_ID VARCHAR(10),
    IN new_ID VARCHAR(10),
    OUT msg VARCHAR(255)
)
BEGIN
    DECLARE exits INT DEFAULT 0;
    DECLARE old_ID_exists INT;
    DECLARE new_ID_exists INT;
    -- 检查旧ID是否存在
    SET old_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = old_ID
        );
    IF old_ID_exists = 0 THEN
        SET msg = '旧ID不存在';
        SET exits = 1;
    END IF;
    -- 检查新ID是否存在
    SET new_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = new_ID
        );
    IF new_ID_exists > 0 THEN
        SET msg = '新ID已存在';
        SET exits = 1;
    END IF;
    -- 没有错误时才开始事务
    IF exits = 0 THEN
        -- 开始事务
        START TRANSACTION;
        -- 获得有旧ID的Reader元组
        -- 将选出的元组中旧ID改为新ID
        -- 将修改过的元组插入Reader表
        INSERT INTO
            Reader (ID, name, age, address)
        SELECT
            new_ID,
            name,
            age,
```

```
            address
        FROM Reader
        WHERE ID = old_ID;
        -- 修改Borrow表中的reader_ID
        UPDATE Borrow
        SET reader_ID = new_ID
        WHERE reader_ID = old_ID;
        -- 修改Reserve表中的reader_ID
        UPDATE Reserve
        SET reader_ID = new_ID
        WHERE reader_ID = old_ID;
        -- 删除旧ID的Reader元组
        DELETE FROM Reader WHERE ID = old_ID;
        -- 提交事务
        COMMIT;
        SET msg = '修改成功';
    END IF;
END;
```

测试 1：把 r1 改成 r2，提示新 ID 已存在



测试 2：把 r1 改成 r1_new，修改成功并且 Borrow 表中的读者 ID 被一并更改



3.2 设计一个存储过程 borrowBook 供读者借书时调用该存储过程完成借书处理

这里的处理是，每本书按照只有 1 本处理。当一个人借成功被预约的书之后，就在预约表中把与这本书相关的预约记录全部删除（由触发器处理），这样就可以保证还书时书的 status 一定设置为 0。

```
CREATE PROCEDURE borrowBook(
    IN Reader_ID CHAR(8),
    IN Book_ID CHAR(8),
    OUT msg VARCHAR(255)
)
BEGIN
    -- 1表示允许借书，0表示不允许借书
    DECLARE borrow_status INT DEFAULT 1;
    -- 1表示借书成功，0表示借书失败
    DECLARE borrow_success INT DEFAULT 1;

    DECLARE Reader_ID_exists INT;
    DECLARE Book_ID_exists INT;
    DECLARE Book_Borrowed INT;
    DECLARE Book_Reserved INT;
    DECLARE Reader_Borrowed_Today INT;
    DECLARE Book_Reserved_By_IN INT;
    DECLARE Reader_Borrowed_Books_Cnt INT;
    DECLARE Book_Reserved_Cnt INT;

    -- 检查读者是否存在
    SET Reader_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = Reader_ID
        );
```

```sql
    IF Reader_ID_exists = 0 THEN
        SET borrow_status = 0;
        SET msg = '读者不存在';
    END IF;
    -- 检查书籍是否存在
    SET Book_ID_exists = (
            SELECT COUNT(*)
            FROM Book
            WHERE ID = Book_ID
        );
    IF Book_ID_exists = 0 THEN
        SET borrow_status = 0;
        SET msg = '书籍不存在';
    END IF;
    -- 检查书是否已经被借出
    SET Book_Borrowed = (
            SELECT COUNT(*)
            FROM Borrow
            WHERE
                Borrow.book_ID = Book_ID
                AND Borrow.return_Date IS NULL
        );
    IF Book_Borrowed > 0 THEN
        SET borrow_status = 0;
        SET msg = '书籍已被借出';
    END IF;
    -- 如果检查合法，才可以开始事务
    IF borrow_status = 1 THEN
        -- 开始事务
        START TRANSACTION;
        -- 1. 查看今天这位读者是否已经借了这本书
        SET Reader_Borrowed_Today = (
            SELECT COUNT(*)
            FROM Borrow
            WHERE
                Borrow.reader_ID = Reader_ID
                AND Borrow.book_ID = Book_ID
                AND DATE(Borrow.borrow_Date) = CURDATE()
            );
        IF Reader_Borrowed_Today > 0 THEN
            SET msg = '今天已经借过这本书';
            SET borrow_success = 0;
        END IF;
        -- 2. 如果这本书已经被预约，而读者不是预约者，则不允许借阅，反之则可以借阅
        --     注意：take_Date如果比今天晚，这样的预约才是有效的
        --     （1）查看书是否在预约状态
        SET Book_Reserved = (
            SELECT COUNT(*)
            FROM Book
            WHERE
                Book.ID = Book_ID
                AND Book.status = 2 -- 被预约
```

```
            );
        --      (2) 查看书是否被读者预约
        SET Book_Reserved_By_IN = (
            SELECT COUNT(*)
            FROM Reserve
            WHERE
                Reserve.book_ID = Book_ID
                AND Reserve.reader_ID = Reader_ID
            );
        IF Book_Reserved_By_IN = 0 AND Book_Reserved > 0 THEN
            SET msg = '这本书已经被预约，而读者不是预约者';
            SET borrow_success = 0;
        END IF;
        -- 3. 如果读者借阅了3本图书且还未归还，则不能借阅
        SET Reader_Borrowed_Books_Cnt = (
            SELECT COUNT(*)
            FROM Borrow
            WHERE
                Borrow.reader_ID = Reader_ID
                AND Borrow.return_Date IS NULL
            );
        IF Reader_Borrowed_Books_Cnt >= 3 THEN
            SET msg = '该读者借阅了3本图书且还未归还';
            SET borrow_success = 0;
        END IF;
        -- 后续处理
        -- 如果借阅成功
        IF borrow_success = 1 THEN
            -- 修改书籍状态为已借出
            UPDATE Book
            SET
                status = 1,
                borrow_Times = borrow_Times + 1
            WHERE ID = Book_ID;
            -- 修改输出值
            SET msg = '借阅成功';
            -- 插入借阅记录
            INSERT INTO Borrow (book_ID, reader_ID, borrow_Date)
            values (Book_ID, Reader_ID, (CURRENT_DATE));
        END IF;
        -- 提交事务
        COMMIT;
    END IF;
END;
```

测试 1：让 r15 借 b10

```
Execute
10  CALL `borrowBook`('r15', 'b10', @msg);
Execute
11  SELECT @msg;  1ms
12
```

Result

搜索结果集 Free 耗时: 1ms < 1 > 共 1 条

	@msg long_blob
1	这本书已经被预约，而读者不是预约者

测试 2：让 r20 借 b10，借阅成功，Book 表状态相应更改，Reserve 的旧记录也被清除

```
Execute
10  CALL `borrowBook`('r20', 'b10', @msg);
Execute
11  SELECT @msg;  1ms
12
```

Book

搜索结果集 Free 耗时: 3ms < 1 > 共 19 条

	* ID char(8)	* name varchar(10)	author varchar(1(	price float	status int	borrow_Times int
1	b1	数据库系统实	Ullman	59	1	8
2	b10	数理逻辑	汪芳庭	22	1	7

Reserve

搜索结果集 Free 耗时: 4ms < 1 > 共 1 条

	* book_ID char(8)	* reader_II char(8)	* reserve_Date date	take_Date date
1	b11	r20	2023-02-17	(NULL)

3.3 设计一个存储过程 returnBook 供读者还书时调用该存储过程完成还书处理

```sql
CREATE PROCEDURE returnBook(
    IN Reader_ID CHAR(8),
    IN Book_ID CHAR(8),
    OUT msg VARCHAR(255)
)
BEGIN
    -- 1表示允许还书，0表示不允许还书
    DECLARE return_status INT DEFAULT 1;
    -- 1表示还书成功，0表示还书失败
    DECLARE return_success INT DEFAULT 1;

    DECLARE Reader_ID_exists INT;
    DECLARE Book_ID_exists INT;
    DECLARE Book_Borrowed INT;

    -- 检查读者是否存在
    SET Reader_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = Reader_ID
        );
    IF Reader_ID_exists = 0 THEN
        SET return_status = 0;
        SET msg = '读者不存在';
    END IF;
    -- 检查书籍是否存在
    SET Book_ID_exists = (
            SELECT COUNT(*)
            FROM Book
            WHERE ID = Book_ID
        );
    IF Book_ID_exists = 0 THEN
        SET return_status = 0;
        SET msg = '书籍不存在';
    END IF;
    -- 检查书是否已经被此人借出
    SET Book_Borrowed = (
            SELECT COUNT(*)
            FROM Borrow
            WHERE
                Borrow.book_ID = Book_ID
                AND Borrow.return_Date IS NULL
                AND Borrow.reader_ID = Reader_ID
        );
    IF Book_Borrowed = 0 THEN
        SET return_status = 0;
        SET msg = '该读者并未借阅这本书';
    END IF;
```

```
    -- 如果检查合法，才可以开始事务
    IF return_status = 1 THEN
        -- 开始事务
        START TRANSACTION;
        -- 修改Book表：这是可以的，因为打算借出之后所有的预约记录都失效
        UPDATE Book
        SET status = 0
        WHERE ID = Book_ID;
        -- 修改Borrow表
        UPDATE Borrow
        SET return_Date = CURRENT_DATE
        WHERE
            Borrow.book_ID = Book_ID
            AND Borrow.return_Date IS NULL
            AND Borrow.reader_ID = Reader_ID;
        -- 修改输出值
        SET msg = '还书成功';
        -- 提交事务
        COMMIT;
    END IF;
END;
```

测试 1：让 r20 还 b10

测试 2：再让 r20 还 b10

3.4 设计一个触发器

前面的存储过程已经用到了这些触发器，故不再单独展示重复的结果。

- 当一本书被预约时，自动将 Book 表中相应图书的 status 修改为 2，并增加 reserve_Times 因为前面没要求实现预约功能，这里先实现 reserveBook 存储过程。

```
CREATE PROCEDURE reserveBook(
    IN Reader_ID CHAR(8),
    IN Book_ID CHAR(8),
    IN Take_Date DATE,
    OUT msg VARCHAR(255)
)
BEGIN
    -- 1表示检查合法，0表示检查不合法
    DECLARE check_legal INT DEFAULT 1;
    DECLARE Reader_ID_exists INT;
    DECLARE Book_ID_exists INT;
    -- 检查读者是否存在
    SET Reader_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = Reader_ID
        );
    IF Reader_ID_exists = 0 THEN
        SET check_legal = 0;
        SET msg = '读者不存在';
    END IF;
    -- 检查书籍是否存在
    SET Book_ID_exists = (
            SELECT COUNT(*)
            FROM Book
            WHERE ID = Book_ID
        );
```

```
    IF Book_ID_exists = 0 THEN
        SET check_legal = 0;
        SET msg = '书籍不存在';
    END IF;
    -- 检查预约日期是否合法
    IF Take_Date <= (CURRENT_DATE) THEN
        SET check_legal = 0;
        SET msg = '预约日期应当晚于当前日期';
    END IF;
    -- 开始事务
    START TRANSACTION;
    IF check_legal = 1 THEN
        -- 向预约表中插入数据
        INSERT INTO Reserve (reader_ID, book_ID, reserve_Date, take_Date)
        value (Reader_ID, Book_ID, (CURRENT_DATE), Take_Date);
        -- 设置msg
        SET msg = '预约成功';
    END IF;
    -- 提交事务
    COMMIT;
END;
```

然后实现触发器。这是后触发器，发生在 Reserve 表产生插入操作之后。

```
CREATE TRIGGER afterReserve
AFTER INSERT ON Reserve
FOR EACH ROW
BEGIN
    UPDATE Book
    SET
        reserve_Times = reserve_Times + 1,
        status = 2
    WHERE ID = NEW.book_ID;
END
```

- 当某本预约的书被借出时或者读者取消预约时，自动减少 reserve_Times

  因为前面也没有要求实现取消预约，所以这里先实现取消预约的过程。

```
CREATE PROCEDURE undoReserveBook(
    IN Reader_ID CHAR(8),
    IN Book_ID CHAR(8),
    OUT msg VARCHAR(255)
)
BEGIN
    -- 1表示检查合法，0表示检查不合法
    DECLARE check_legal INT DEFAULT 1;
    DECLARE Reader_ID_exists INT;
    DECLARE Book_ID_exists INT;
    DECLARE tuple_reserve_Date DATE;
    DECLARE tuple_take_Date DATE;
    DECLARE Is_Reader_Reserve INT;
    -- 检查读者是否存在
```

```sql
    SET Reader_ID_exists = (
            SELECT COUNT(*)
            FROM Reader
            WHERE ID = Reader_ID
        );
    IF Reader_ID_exists = 0 THEN
        SET check_legal = 0;
        SET msg = '读者不存在';
    END IF;
    -- 检查书籍是否存在
    SET Book_ID_exists = (
            SELECT COUNT(*)
            FROM Book
            WHERE ID = Book_ID
        );
    IF Book_ID_exists = 0 THEN
        SET check_legal = 0;
        SET msg = '书籍不存在';
    END IF;
    -- 检查该读者是否预约了该书籍
    SET Is_Reader_Reserve = (
        SELECT COUNT(*)
        FROM Reserve
        WHERE reader_ID = Reader_ID AND book_ID = Book_ID
    );
    IF Is_Reader_Reserve = 0 THEN
        SET check_legal = 0;
        SET msg = '该读者没有预约该书籍';
    END IF;
    -- 开始事务
    START TRANSACTION;
    IF check_legal = 1 THEN
        -- 删除预约表最新的关于该读者和该书籍的记录
        SET tuple_reserve_Date = (
            SELECT reserve_Date
            FROM Reserve
            WHERE reader_ID = Reader_ID AND book_ID = Book_ID
            ORDER BY reserve_Date DESC
            LIMIT 1
        );
        SET tuple_take_Date = (
            SELECT take_Date
            FROM Reserve
            WHERE reader_ID = Reader_ID AND book_ID = Book_ID
            ORDER BY reserve_Date DESC
            LIMIT 1
        );
        DELETE FROM Reserve
        WHERE
            reader_ID = Reader_ID
            AND book_ID = Book_ID
            AND reserve_Date = tuple_reserve_Date
```

```
            AND take_Date = tuple_take_Date;
        -- 设置msg
        SET msg = '取消预约成功';
    END IF;
    -- 提交事务
    COMMIT;
END;
```

实现借书之后的触发器:

```
CREATE TRIGGER afterBorrow
BEFORE UPDATE ON Book FOR EACH ROW
BEGIN
    -- 因为我的实现逻辑
    -- 借书之后需要删除若干预约记录
    DECLARE reserved_cnt INT;
    IF NEW.borrow_Times > OLD.borrow_Times THEN
        -- 计算需要删除的预约记录数
        SET reserved_cnt = (
            SELECT COUNT(*) FROM Reserve
            WHERE book_ID = NEW.ID
        );
        -- 从预约表中删除这些记录
        DELETE FROM Reserve
        WHERE book_ID = NEW.ID;
        -- 减少Book表的reserved_Times字段
        SET NEW.reserve_Times = NEW.reserve_Times - reserved_cnt;
    END IF;
END
```

实现取消预约之后的触发器:

```
CREATE TRIGGER afterUndoReserve
AFTER DELETE ON Reserve
FOR EACH ROW
BEGIN
    UPDATE Book
    SET
        reserve_Times = reserve_Times - 1
    WHERE ID = OLD.book_ID;
END
```