浙江大学

数据库系统实验报告

作业名称：	SQL 数据完整性
姓　　名：	龙永奇
学　　号：	3220105907
电子邮箱：	3220105907@zju.edu.cn
联系电话：	15393113093
指导老师：	孙建伶

2024 年 03 月 27 日

SQL 数据完整性

一、实验目的

1. 熟悉通过 SQL 进行数据完整性控制的方法

二、实验环境

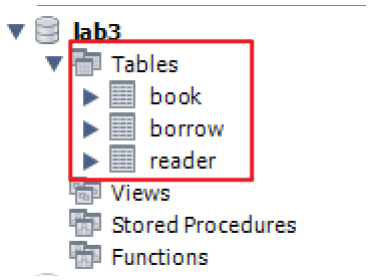
1. 数据库管理系统：MySQL
2. 操作系统：Windows11 企业版

三、实验流程

1. 定义若干表，其中包括 primary key, foreign key 和 check 的定义

```sql
CREATE DATABASE LAB3;
USE LAB3;
CREATE TABLE book(
    book_id INT,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    publication_year YEAR,
    ISBN VARCHAR(6),

    PRIMARY KEY(book_id),
    CHECK(LENGTH(ISBN) = 6)
);

CREATE TABLE reader(
    reader_id INT,
    reader_name VARCHAR(50) NOT NULL,
    phone_num VARCHAR(11) UNIQUE,

    PRIMARY KEY(reader_id),
);

CREATE TABLE borrow(
    borrow_id INT,
    book_id INT,
```

```
    reader_id INT,
    borrow_date DATE NOT NULL,
    return_date DATE,

    PRIMARY KEY(borrow_id),
    Foreign Key (book_id) REFERENCES book(book_id),
    Foreign Key (reader_id) REFERENCES reader(reader_id),
    CHECK (return_date >= borrow_date)
);
```



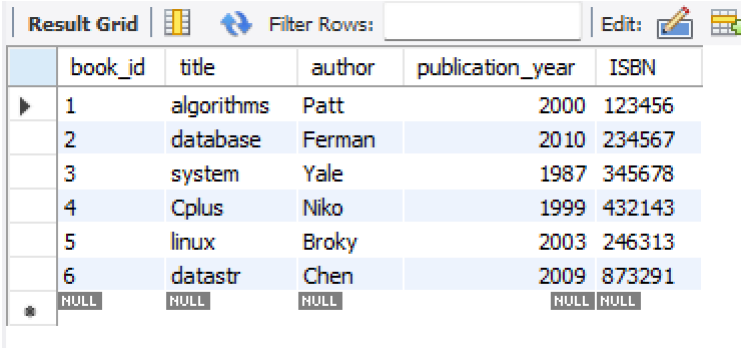
本次实验使用数据库管理图书的借阅，其中 book 表示藏书，reader 表示读者，borrow 表示借阅信息

2. 让表中插入数据，考察 primary key 如何控制实体完整性

```
INSERT INTO book(book_id, title, author, publication_year, ISBN)
VALUES
    (1, 'algorithms', 'Patt', 2000, '123456'),
    (2, 'database', 'Ferman', 2010, '234567'),
    (3, 'system', 'Yale', 1987, '345678'),
    (4, 'Cplus', 'Niko', 1999, '432143'),
    (5, 'linux', 'Broky', 2003, '246313'),
    (6, 'datastr', 'Chen', 2009, '873291'),

INSERT INTO reader(reader_id, reader_name, phone_num)
VALUES
    (1, 'Mary', '37821739199'),
    (2, 'Jack', '81832713989'),
    (3, 'Tomy', '21639213890'),
    (4, 'Smit', '98327182638');

INSERT INTO borrow(borrow_id, book_id, reader_id, borrow_date,
return_date)
VALUES
```

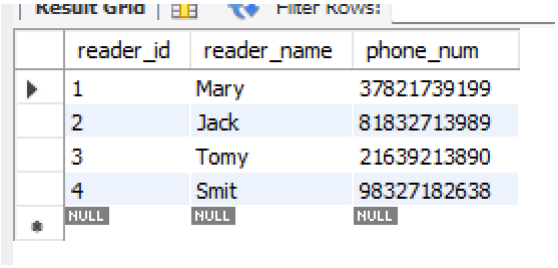
```
    (1, 4, 1, '2024-03-01', '2024-03-15'),
    (2, 2, 2, '2024-03-05', NULL),
    (3, 5, 3, '2024-03-10', '2024-03-20')
    (4, 6, 3, '2024-03-01', '2024-03-15'),
    (5, 6, 4, '2024-03-05', NULL),
    (6, 3, 4, '2024-03-10', '2024-03-20');
```

插入结果如下：



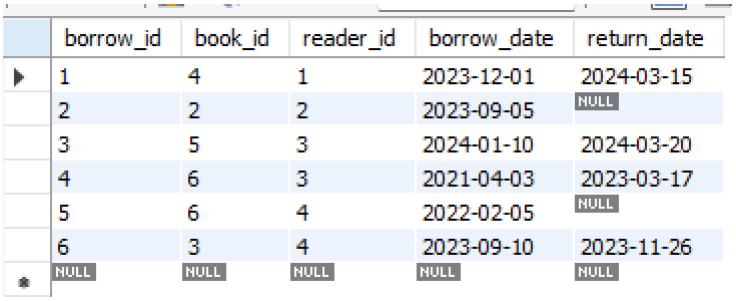
book_id	title	author	publication_year	ISBN
1	algorithms	Patt	2000	123456
2	database	Ferman	2010	234567
3	system	Yale	1987	345678
4	Cplus	Niko	1999	432143
5	linux	Broky	2003	246313
6	datastr	Chen	2009	873291
NULL	NULL	NULL	NULL	NULL

book



reader_id	reader_name	phone_num
1	Mary	37821739199
2	Jack	81832713989
3	Tomy	21639213890
4	Smit	98327182638
NULL	NULL	NULL

reader



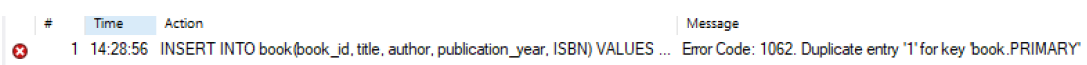
borrow_id	book_id	reader_id	borrow_date	return_date
1	4	1	2023-12-01	2024-03-15
2	2	2	2023-09-05	NULL
3	5	3	2024-01-10	2024-03-20
4	6	3	2021-04-03	2023-03-17
5	6	4	2022-02-05	NULL
6	3	4	2023-09-10	2023-11-26
NULL	NULL	NULL	NULL	NULL

borrow

插入一条主键 book_id 重复的记录：

```
INSERT INTO book(book_id, title, author, publication_year, ISBN)
VALUES
    (1, 'compiler', 'Say', 1990, '288341');
```

出现错误：

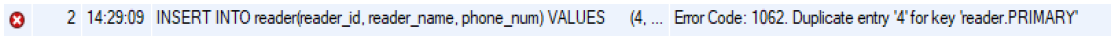


#	Time	Action	Message
1	14:28:56	INSERT INTO book(book_id, title, author, publication_year, ISBN) VALUES ...	Error Code: 1062. Duplicate entry '1' for key 'book.PRIMARY'

插入一条主键 reader_id 重复的记录：

```
INSERT INTO reader(reader_id, reader_name, phone_num)
VALUES
    (4, 'boot', '38912089021');
```

亦会出现错误：

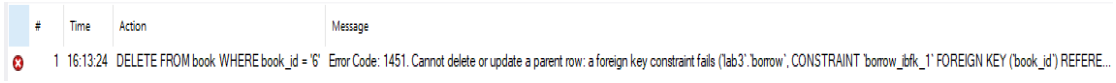


3. 删除被引用表中的行，考察 foreign key 中 on delete 子句如何控制参照完整性。

删除 book 中 book_id 为 1 的元组：

```
DELETE FROM book
WHERE book_id = '6';
```

出现错误：

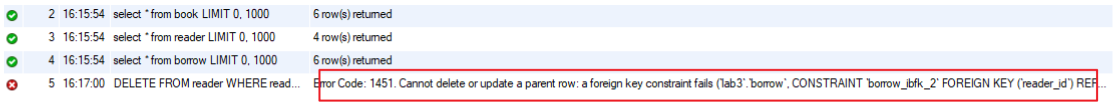


由于在实验二时已经将数据库中的 safe mode 关闭，因此并未出现实验指导中的错误

删除 reader 中 reader_id 为 4 的元组：

```
DELETE FROM reader
WHERE reader_id = '4'
```

出现错误：



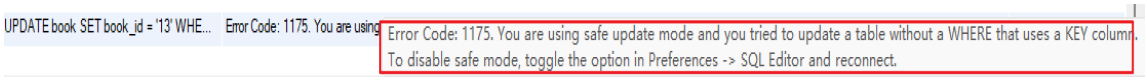
说明 borrow 表中的两个外键均起到保护数据库完整性的作用，体现 on delete 字句如何控制参照完整性

4. 修改被引用表中的行的 primary key，考察 foreign key 中 on update 子句如何控制参照完整性。

修改 book 中 title 为 algorithms 的 book_id：

```
UPDATE book
SET book_id = '13'
WHERE title = 'algorithms';
```

出现错误

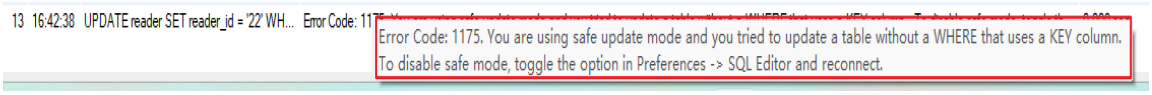


修改 reader 中 reader_name 为 Tomy 的 reader_id：

```
UPDATE reader
```

```
SET reader_id = '22'
WHERE reader_name = 'Tomy';
```

仍出现错误：

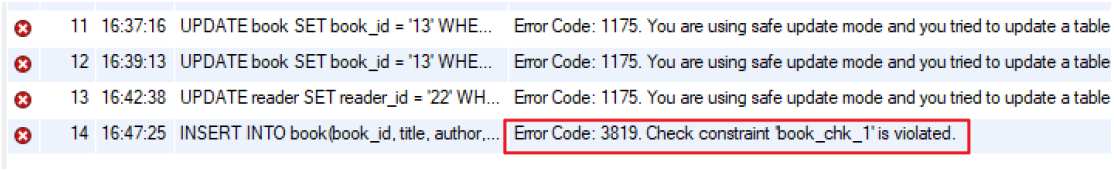


5. 修改或插入表中数据，考察 check 子句如何控制校验完整性。

在 book 中插入不符合 check 要求的 ISBN 长度的元组：

```
INSERT INTO book(book_id, title, author, publication_year, ISBN)
VALUES
    (7, 'spaceX', 'Musk', 2001, '222');
```

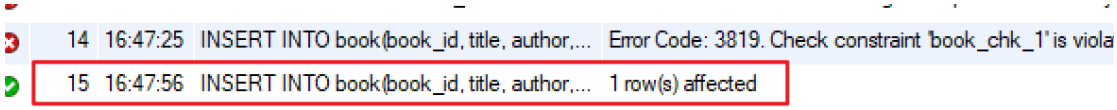
出现错误



如果改为：

```
INSERT INTO book(book_id, title, author, publication_year, ISBN)
VALUES
    (7, 'spaceX', 'Musk', 2001, '222333');
```

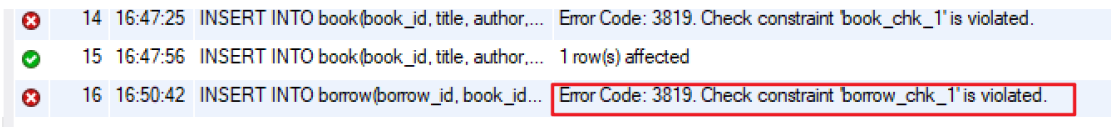
则插入成功：



在 borrow 中插入不符合 check 要求的 borrow_date 和 return_date：

```
INSERT INTO borrow(borrow_id, book_id, reader_id, borrow_date,
return_date)
VALUES
    (7, 1, 1, '2024-03-01', '2021-03-15');
```

仍然出现错误：



综上，说明 check 对表内元组起到了约束作用

6. 定义一个 assertion，并通过修改表中数据考察断言如何控制数据完整性。

规定每人借书数目小于 50 本：

```
CREATE ASSERTION borrow_upbound
CHECK(NOT EXISTS(
    SELECT reader_id, COUNT(book_id) AS num_borrowed_books
    FROM borrow
    WHERE num_borrowed_books > 50
    GROUP BY reader_id)
    );
```

MySQL 中并不支持全局断言：

```
CREATE ASSERTION borrow_upbound
CHECK(NOT EXISTS(
    SELECT reader_id, COUNT(book_id) AS num_borrowed_books
    FROM borrow
    WHERE num_borrowed_books > 50
    GROUP BY reader_id;)
    );
```

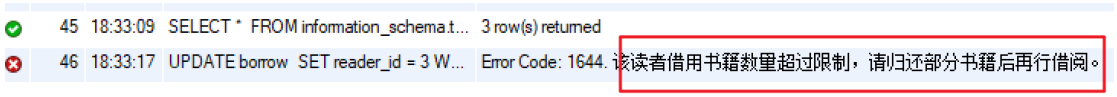
7. 定义一个 trigger，并通过修改表中数据考察触发器如何起作用。

定义触发器：

```
DELIMITER //
CREATE TRIGGER borrow_limit
AFTER UPDATE ON borrow
FOR EACH ROW
BEGIN
    DECLARE borrow_nums INT;

    SELECT COUNT(*) INTO borrow_nums
    FROM borrow
    WHERE reader_id = NEW.reader_id;

    IF borrow_nums > 2 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '该读者借用书籍数量超过限制，请归还部分书籍后再行借阅。';
    END IF;
END; //
```

如果一个人借用超过 2 本书，则不予借用：

```
UPDATE borrow
SET reader_id = '3'
WHERE borrow_id = '2';
```

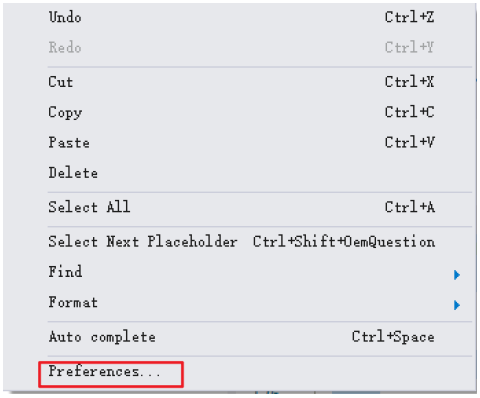


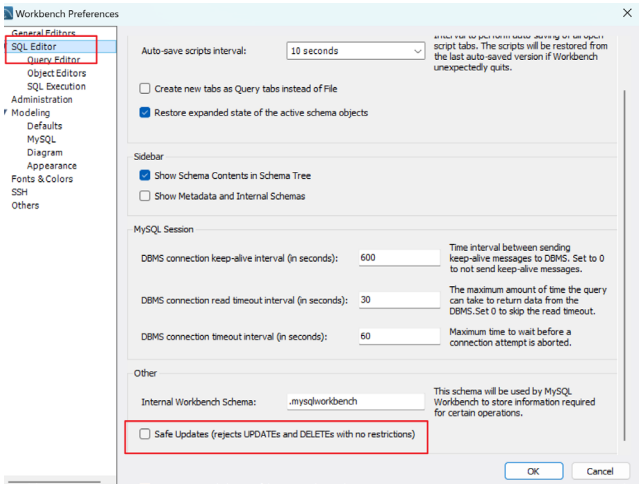
四、遇到的问题及解决方法

1. 在步骤 3 删除引用表中的行来验证 foreign key 中 on delete 如何控制参照完整性的过程中未出现安全模式的类似问题：

   You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.0.000 sec

   但是在步骤 4 更新引用表中的行来验证 foreign key 中 on update 如何控制参照完整性的过程中仍然出现该问题，因此在 Preferences 中关闭了 safe mode：





2. 步骤 6 无法使用全局断言，全局断言在 MySQL 不支持，经过查询，发现几乎没有数据库支持全局断言，此功能实现成本较大，限制较多
3. 步骤 7 定义触发器的过程中出现了一个语句多个 ； 的问题，通过 DELIMITER 得以解决

五、总结

在本次实验中，我们通过创建表、插入数据、定义约束、触发器等操作，熟悉了通过 SQL 进行数据完整性控制的相关方法：主键约束确保每行数据具有唯一标识，外键约束建立表之间的关联关系，检查约束限制数据的取值范围或格式。触发器能够在数据变化时执行额外逻辑，异常处理机制则帮助我们处理数据操作中的错误或异常情况。通过这些约束，我们可以确保数据库的数据安全性和一致性，为应用程序的稳定运行提供了基础保障。