第五次实验

> 22375080
>
> 杨佳宇轩

Task 1

1-1

使用存储过程 + 游标

```
-- 用户密码检查/修改： 接收四个参数（用户名，密码，新密码，动作），若动作为1，则检查用户名和密码是否和密码表中存的相符，相符则返回 true，不相符返回false；若动作为2， 则首先检查用户名、密码是否相符，若不相符返回false，相符则将密码表中的密码改成新密码，返回true。密码要求只包含数字和字母，长度大于等于4、小于等于10；
DELIMITER $$
CREATE PROCEDURE sp_password(
    IN p_username VARCHAR(50),
    IN p_password VARCHAR(100),
    IN p_new_password VARCHAR(100),
    IN p_action TINYINT,
    OUT p_result BOOLEAN
)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE cur_user VARCHAR(50);
    DECLARE cur_pwd VARCHAR(100);
    DECLARE acct_cursor CURSOR FOR
        SELECT username, password FROM accounts;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1; -- 异常处理

    IF p_action = 1 THEN
        OPEN acct_cursor;
        read_loop: LOOP
            FETCH acct_cursor INTO cur_user, cur_pwd;
            IF done = 1 THEN
                SET p_result = FALSE;
                LEAVE read_loop;
            END IF;
```

```
            IF cur_user = p_username AND cur_pwd = p_password
THEN
                SET p_result = TRUE;
                CLOSE acct_cursor;
                LEAVE read_loop;
            END IF;
        END LOOP;
    ELSEIF p_action = 2 THEN
            IF NOT EXISTS (
            SELECT 1 FROM accounts
             WHERE username = p_username
               AND password = p_password
        ) THEN
            SET p_result = FALSE;
        ELSEIF LENGTH(p_new_password) < 4
              OR LENGTH(p_new_password) > 10
              OR p_new_password REGEXP '[^a-zA-Z0-9]'
        THEN
            SET p_result = FALSE;
        ELSE
            UPDATE accounts
               SET password = p_new_password
             WHERE username = p_username;
            SET p_result = TRUE;
        END IF;
    ELSE
        SET p_result = FALSE;
    END IF;
END$$
DELIMITER ;
```

测试

```
CALL sp_password('alice','alice_pass','',          1, @r1);
SELECT @r1 AS `result`;      -- TRUE
CALL sp_password('alice','wrong_pass','',          1, @r2);
SELECT @r2 AS `result`;            -- FALSE
CALL sp_password('alice','alice_pass','newPwd1',   2, @r3);
SELECT @r3 AS `result`;-- TRUE
```

结果

result
1

result
0

result
1

1-2

```
-- 借书：接收两个参数（用户名，ISBN），没有足够的书、用户不存在或一个人借阅两本同样的书时返回false，合法执行后，借阅记录表会新增一条记录，书库对应书的数量也需要减1，并返回true；
DROP PROCEDURE IF EXISTS sp_borrow_book;
DELIMITER $$
CREATE PROCEDURE sp_borrow_book(
  IN  p_username VARCHAR(50),
  IN  p_ISBN     VARCHAR(20),
  OUT p_result   TINYINT(1)
)
BEGIN
  SET p_result = 0;

  DECLARE v_user_exists    INT DEFAULT 0;
  DECLARE v_book_count     INT DEFAULT 0;
  DECLARE v_already_borrow INT DEFAULT 0;

  SELECT COUNT(*) INTO v_user_exists
    FROM accounts
```

```
    WHERE username = p_username;

  IF v_user_exists > 0 THEN

    SELECT `count` INTO v_book_count
      FROM books
     WHERE ISBN = p_ISBN;

    IF v_book_count IS NOT NULL
       AND v_book_count >= 1 THEN

      SELECT COUNT(*) INTO v_already_borrow
        FROM borrow_records
       WHERE username   = p_username
         AND ISBN       = p_ISBN
         AND back_time IS NULL;

      IF v_already_borrow = 0 THEN

        INSERT INTO borrow_records(username, ISBN, begin_time)
             VALUES(p_username, p_ISBN, CURDATE());
        UPDATE books
           SET `count` = `count` - 1
         WHERE ISBN = p_ISBN;

        SET p_result = 1;

      END IF;

    END IF;

  END IF;

END$$

DELIMITER ;
```

测试

```
CALL sp_borrow_book('alice','978-0-00-000000-1', @b1); SELECT @b1
AS `result`;    -- TRUE
CALL sp_borrow_book('alice','978-0-00-000000-1', @b2); SELECT @b2
AS `result`;                -- FALSE
CALL sp_borrow_book('nonuser','978-0-00-000000-2', @b3); SELECT
@b3 AS `result`;         -- FALSE
UPDATE books SET `count`=0 WHERE ISBN='978-0-00-000000-3';
CALL sp_borrow_book('bob','978-0-00-000000-3', @b4);   SELECT @b4
AS `result`;               -- FALSE
```

1-3

```
DELIMITER $$
CREATE PROCEDURE sp_return_book(
    IN  p_username VARCHAR(50),
    IN  p_ISBN     VARCHAR(20),
    OUT p_result   BOOLEAN
)
BEGIN
        DECLARE done INT DEFAULT 0;
    DECLARE v_rec_exists INT DEFAULT 0;

    SELECT COUNT(*) INTO v_rec_exists
      FROM borrow_records
     WHERE username = p_username
       AND ISBN     = p_ISBN
       AND back_time IS NULL;
    IF v_rec_exists = 0 THEN
        SET p_result = FALSE;
                SET done = 1;
    END IF;

        IF done = 0 THEN
            UPDATE borrow_records
                SET back_time = CURDATE()
            WHERE username = p_username
                AND ISBN     = p_ISBN
                AND back_time IS NULL;
            UPDATE books
                SET `count` = `count` + 1
            WHERE ISBN = p_ISBN;
            SET p_result = TRUE;
        END IF;
END$$
```

```
DELIMITER ;
```

测试

```
CALL sp_return_book('alice','978-0-00-000000-2', @rtn1); SELECT
@rtn1 AS `alice 还未借书`;       -- FALSE
CALL sp_return_book('alice','978-0-00-000000-1', @rtn2); SELECT
@rtn2 AS `alice 正常还书`;       -- TRUE
CALL sp_return_book('alice','978-0-00-000000-1', @rtn3); SELECT
@rtn3 AS `重复还书`;           -- FALSE
CALL sp_return_book('ghost','978-0-00-000000-3', @rtn4); SELECT
@rtn4 AS `不存在用户还书`;     -- FALSE
```

结果

alice 还未借书
0

alice 正常还书
1

重复还书
0

不存在用户还书
0

1-4

```
-- 查看当前借阅记录：接受一个参数（用户名），返回该用户名的当前借阅中的记录(用户名, ISBN, 到期时间)
DELIMITER $$
CREATE PROCEDURE sp_view (
    IN p_username VARCHAR(50)
)
BEGIN
    SELECT username, ISBN, end_time
    FROM borrow_records
    WHERE username = p_username
        AND back_time IS NULL;
END$$
DELIMITER ;
```

测试

```
CALL sp_view('alice');
CALL sp_view('bob');
CALL sp_view('charlie');
```

结果

username	ISBN	end_time
(N/A)	(N/A)	(N/A)

username	ISBN	end_time
(N/A)	(N/A)	(N/A)

username	ISBN	end_time
charlie	978-0-00-000000-3	2025-05-05

Task 2

设置 `PRUCEDURE` 用户更新单个用户

```
DELIMITER $$
CREATE PROCEDURE update_customer_level(IN qid INT)
BEGIN
    DECLARE total INT DEFAULT 0;
    SELECT SUM(f.price * s.quantity) INTO total
        FROM sells as s
        JOIN fruit as f ON s.fid = f.fid
        WHERE s.cid = qid;
    IF total > 20000 THEN
        UPDATE customer SET level = 'SVIP' WHERE cid = qid;
    ELSEIF total > 10000 THEN
        UPDATE customer SET level = 'VIP' WHERE cid = qid;
    ELSE
        UPDATE customer SET level = 'normal' WHERE cid = qid;
    END IF;
END$$
DELIMITER ;
```

设置 `TRIGGER`

```
-- TRIGGER INSERT
DELIMITER $$
CREATE TRIGGER trg_insert
AFTER INSERT ON sells
FOR EACH ROW
```

```
BEGIN
    CALL update_customer_level(NEW.cid);
END$$
DELIMITER ;

-- TRIGGER UPDATE
DROP TRIGGER IF EXISTS trg_sells_after_update;
DELIMITER $$
CREATE TRIGGER trg_sells_after_update
AFTER UPDATE ON sells
FOR EACH ROW
BEGIN
  IF OLD.cid <> NEW.cid THEN
    CALL update_customer_level(OLD.cid);
    CALL update_customer_level(NEW.cid);
  ELSE
    CALL update_customer_level(NEW.cid);
  END IF;
END$$
DELIMITER ;

-- TRIGGER DELETE
DROP TRIGGER IF EXISTS trg_sells_after_delete;
DELIMITER $$
CREATE TRIGGER trg_sells_after_delete
AFTER DELETE ON sells
FOR EACH ROW
BEGIN
  CALL update_customer_level(OLD.cid);
END$$
DELIMITER ;
```

测试

```
-- 5.1 查看初始客户等级
SELECT * FROM customer;

-- 5.2 插入一条价值 9,000 的购买记录（应为 normal）
INSERT INTO sells VALUES (1, 100, '2025-04-15 10:00:00', 1800);
 -- 5.00 * 1800 = 9000
SELECT cid, cname, level FROM customer WHERE cid = 100;

-- 5.3 再插入一条价值 3,000 的购买记录（累计 12,000，应升级为 VIP）
INSERT INTO sells VALUES (2, 100, '2025-04-16 11:00:00', 857);
-- 3.50 * 857 ≈ 3000
```

```
SELECT cid, cname, level FROM customer WHERE cid = 100;

-- 5.4 修改上一条记录的数量为 6000（额外约 21,000，总计 30,000，应升级为 SVIP）
UPDATE sells
  SET quantity = 6000
  WHERE fid = 2 AND cid = 100 AND sellTime = '2025-04-16 11:00:00';
SELECT cid, cname, level FROM customer WHERE cid = 100;

-- 5.5 删除第一条记录（去掉 9,000，剩余 ≈ 21,000，应仍为 SVIP）
DELETE FROM sells
  WHERE fid = 1 AND cid = 100 AND sellTime = '2025-04-15 10:00:00';
SELECT cid, cname, level FROM customer WHERE cid = 100;

-- 5.6 删除第二条记录（去掉约 21,000，剩余 0，应降回 normal）
DELETE FROM sells
  WHERE fid = 2 AND cid = 100 AND sellTime = '2025-04-16 11:00:00';
SELECT cid, cname, level FROM customer WHERE cid = 100;
```

结果

cid	cname	level
100	张三	normal
101	李四	normal
102	王五	normal

cid	cname	level
100	张三	normal

cid	cname	level
100	张三	VIP

cid	cname	level
100	张三	SVIP

cid	cname	level
100	张三	SVIP

cid	cname	level
100	张三	normal

附录

Task 1 数据初始化

```
-- 1. 创建账户表
CREATE TABLE accounts (
  username   VARCHAR(50) PRIMARY KEY,
  password   VARCHAR(100) NOT NULL
);

-- 2. 创建书库表
CREATE TABLE books (
  ISBN       VARCHAR(20) PRIMARY KEY,
  name       VARCHAR(100) NOT NULL,
  count      INT          NOT NULL
);

-- 3. 创建借阅记录表（到期时间通过触发器自动设为借书时间+30天）
CREATE TABLE borrow_records (
  username    VARCHAR(50) NOT NULL,
  ISBN        VARCHAR(20) NOT NULL,
  begin_time  DATE        NOT NULL,
  end_time    DATE        NOT NULL,
  back_time   DATE        NULL,
  PRIMARY KEY (username, ISBN, begin_time),
  FOREIGN KEY (username) REFERENCES accounts(username),
  FOREIGN KEY (ISBN)     REFERENCES books(ISBN)
);

-- 4. 触发器：插入新记录时自动计算 end_time = begin_time + 30 天
DELIMITER $$
CREATE TRIGGER trg_set_end_time
BEFORE INSERT ON borrow_records
```

```
FOR EACH ROW
BEGIN
  SET NEW.end_time = DATE_ADD(NEW.begin_time, INTERVAL 30 DAY);
END$$
DELIMITER ;

-- accounts 表
INSERT INTO accounts (username, password) VALUES
('alice',   'alice_pass'),
('bob',     'bob_pass'),
('charlie', 'charlie_pass');

-- books 表
INSERT INTO books (ISBN,         name,                count) VALUES
('978-0-00-000000-1', '《数据库系统概论》',    5),
('978-0-00-000000-2', '《算法导论》',          3),
('978-0-00-000000-3', '《计算机网络》',        4);

-- borrow_records 表
-- alice 于 2025-04-01 借《数据库系统概论》，未还
INSERT INTO borrow_records (username, ISBN, begin_time,
back_time) VALUES
('alice',   '978-0-00-000000-1', '2025-04-01',   NULL);

-- bob 于 2025-03-20 借《算法导论》，于 2025-04-10 归还
INSERT INTO borrow_records (username, ISBN, begin_time,
back_time) VALUES
('bob',     '978-0-00-000000-2', '2025-03-20',   '2025-04-10');

-- charlie 于 2025-04-05 借《计算机网络》，未还
INSERT INTO borrow_records (username, ISBN, begin_time,
back_time) VALUES
('charlie', '978-0-00-000000-3', '2025-04-05',   NULL);
```