哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

实验二 高级SQL语言的使用

2023秋

本学期实验总体安排

本学期实验课程共 16 个学时，4 个实验项目，总成绩为 30 分。

实验项目	实验一	实验二	实验三		实验四
学时	2	2	4	4	4
实验内容	MySQL及SQL的使用	高级SQL的使用	一个小型系统的设计与实现		查询处理算法的模拟实现
分数	5	5	12		8

目录

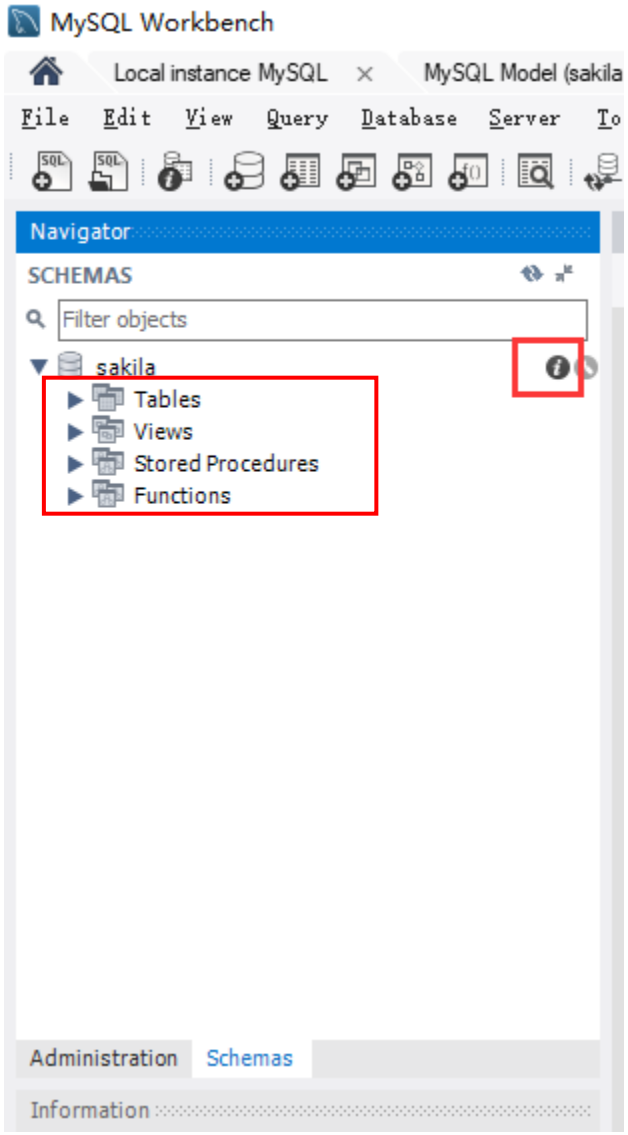
实验目的

1. 理解视图、触发器、约束、存储过程和函数的基本概念，掌握它们的用法；
2. 能结合实例设计合适的视图、触发器和存储过程；
3. 结合实验加深对数据库完整性和安全性的理解。

实验内容

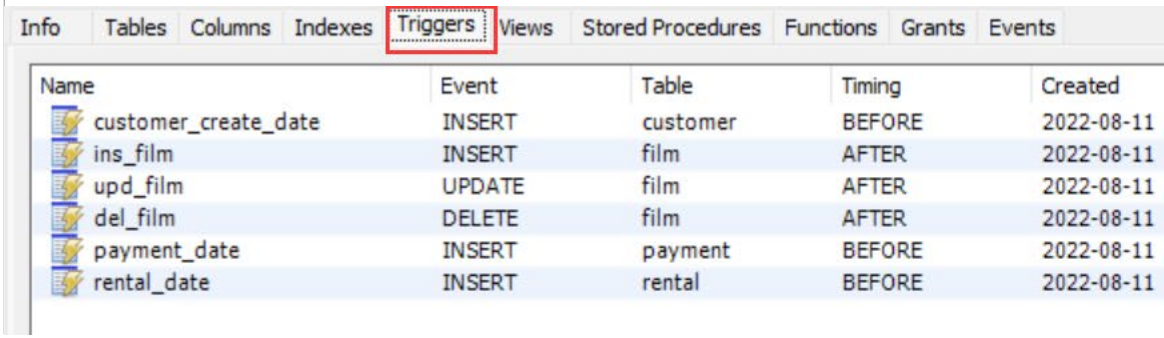
1. 理解和分析Sakila数据库中的视图、触发器、约束、存储过程和函数；
2. 根据场景，为Sakila数据库设计并实现合理的视图、触发器和存储过程；
3. 创建新的数据库用户，并为其分配权限。

实验原理



Sakila样例数据库包括：

- 16 张表
- 7个视图
- 6个触发器
- 3个存储过程
- 3个函数

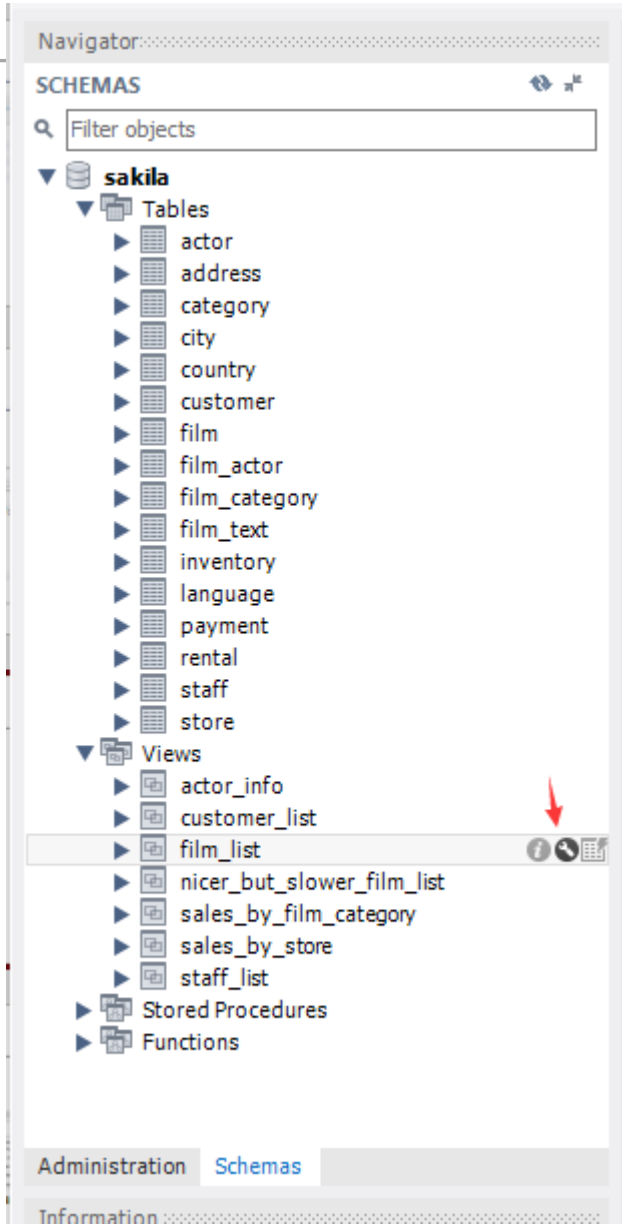


Name	Event	Table	Timing	Created
customer_create_date	INSERT	customer	BEFORE	2022-08-11
ins_film	INSERT	film	AFTER	2022-08-11
upd_film	UPDATE	film	AFTER	2022-08-11
del_film	DELETE	film	AFTER	2022-08-11
payment_date	INSERT	payment	BEFORE	2022-08-11
rental_date	INSERT	rental	BEFORE	2022-08-11

实验原理

视图（Views）

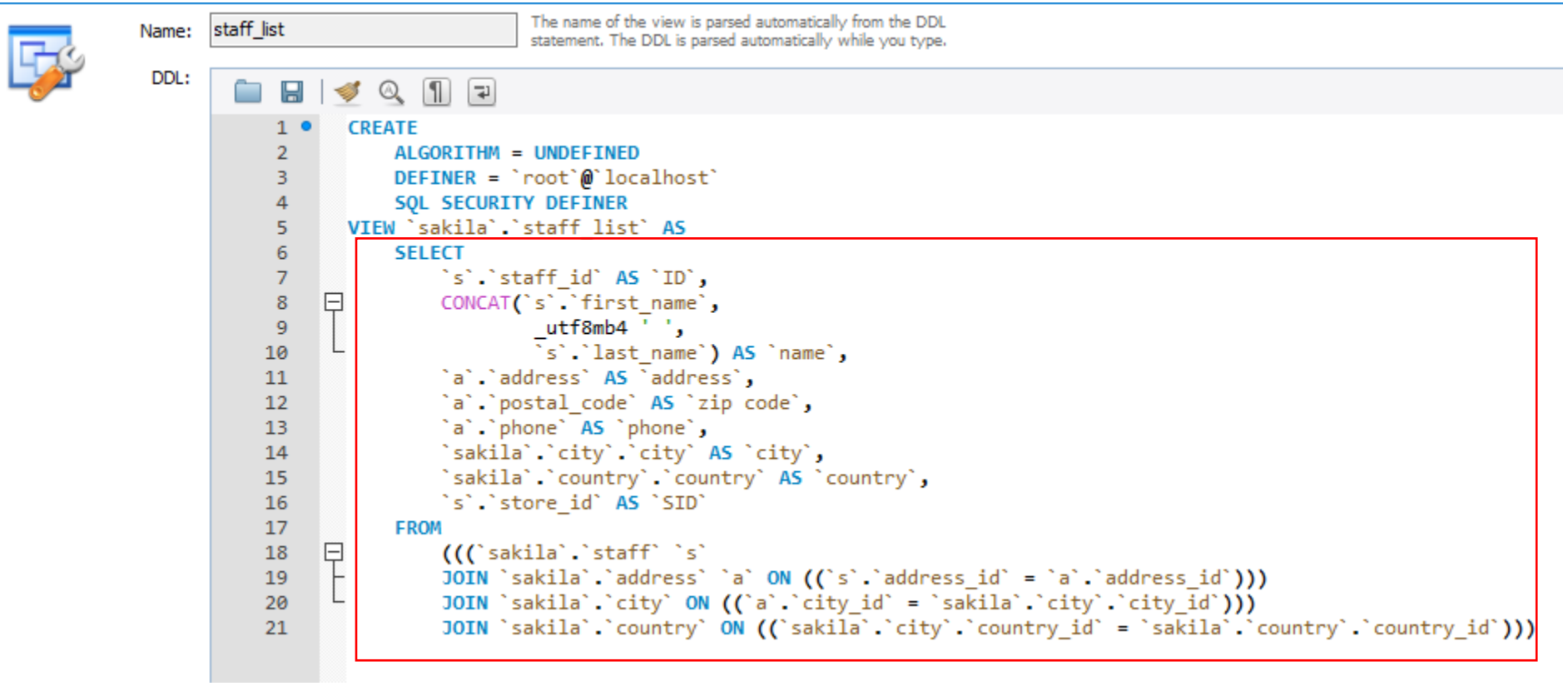
- 简单
- 安全
- 数据独立



实验原理

视图（Views）

staff_list




```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sakila`.`staff_list` AS
    SELECT
        `s`.`staff_id` AS `ID`,
        CONCAT(`s`.`first_name`,
                _utf8mb4 ' ',
                `s`.`last_name`) AS `name`,
        `a`.`address` AS `address`,
        `a`.`postal_code` AS `zip code`,
        `a`.`phone` AS `phone`,
        `sakila`.`city`.`city` AS `city`,
        `sakila`.`country`.`country` AS `country`,
        `s`.`store_id` AS `SID`
    FROM
        (((`sakila`.`staff` `s`
        JOIN `sakila`.`address` `a` ON ((`s`.`address_id` = `a`.`address_id`)))
        JOIN `sakila`.`city` ON ((`a`.`city_id` = `sakila`.`city`.`city_id`)))
        JOIN `sakila`.`country` ON ((`sakila`.`city`.`country_id` = `sakila`.`country`.`country_id`)))
```

实验原理

视图（Views）

```
1  SELECT
2      `s`.`staff_id` AS `ID`,
3      CONCAT(`s`.`first_name`,
4              _utf8mb4 ' ',
5              `s`.`last_name`) AS `name`,
6      `a`.`address` AS `address`,
7      `a`.`postal_code`
8      `a`.`phone` AS `ph
9      `city`.`city` AS `
10     `country`.`country
11     `s`.`store_id` AS
12 FROM
13     (((`staff` `s`
14     JOIN `address` `a`
15     JOIN `city` ON ((`
16     JOIN `country` ON
```

Limit to 2000 rows

```
1  select * from staff_list;
2
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

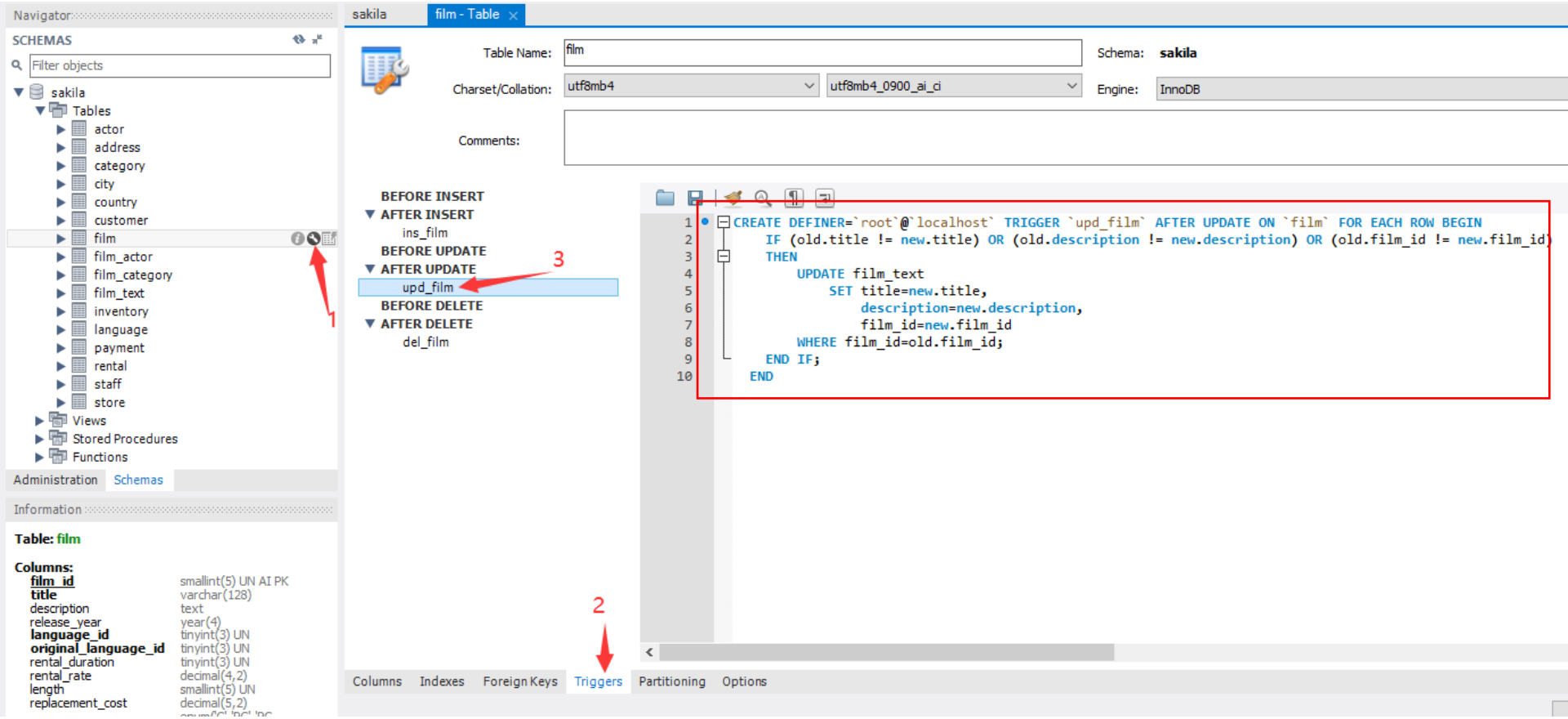
ID	name	address	zip code	phone	city	country	SID
1	Mike Hillyer	23 Workhaven Lane		14033335568	Lethbridge	Canada	1
2	Jon Stephens	1411 Lillydale Drive		6172235589	Woodridge	Australia	2

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	address	zip code	phone	city	country	SID
1	Mike Hillyer	23 Workhaven Lane		14033335568	Lethbridge	Canada	1
2	Jon Stephens	1411 Lillydale Drive		6172235589	Woodridge	Australia	2

实验原理

触发器（Triggers）



实验原理

触发器（Triggers）

```
CREATE DEFINER=`root`@`localhost` TRIGGER `upd_film` AFTER UPDATE ON `film` FOR EACH ROW BEGIN
    IF (old.title != new.title) OR (old.description != new.description) OR (old.film_id != new.film_id)
    THEN
        UPDATE film_text
            SET title=new.title,
                description=new.description,
                film_id=new.film_id
        WHERE film_id=old.film_id;
    END IF;
END
```

触发器创建语法四要素：

① 监视点(table)

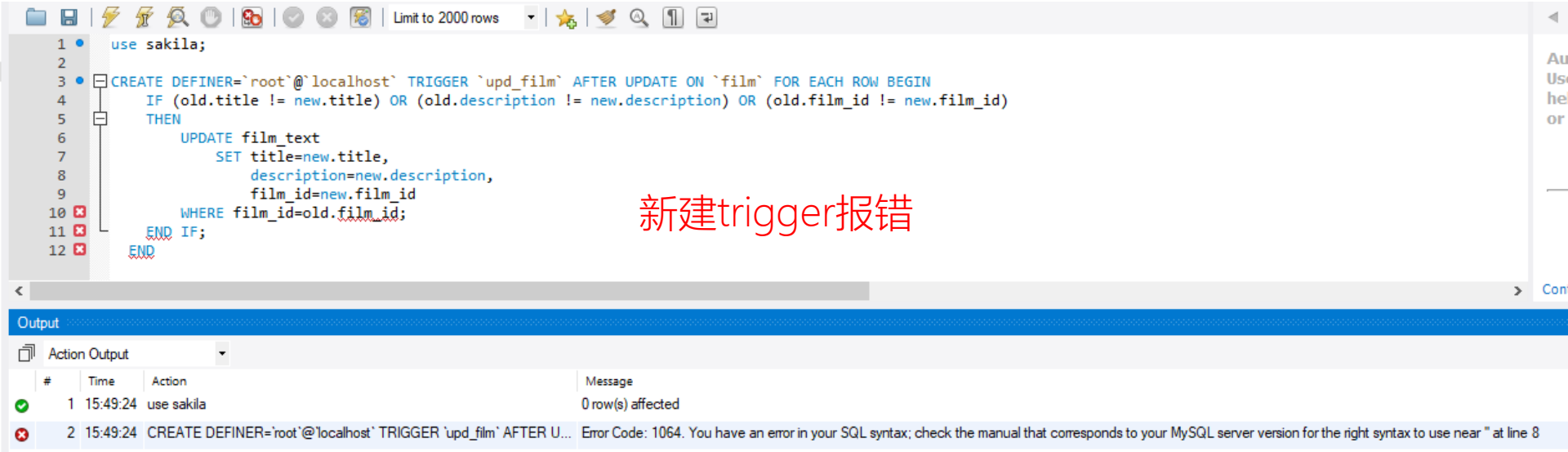
② 监视事件(insert/update/delete)

③ 触发时间(after/before)

④ 触发事件(insert/update/delete)

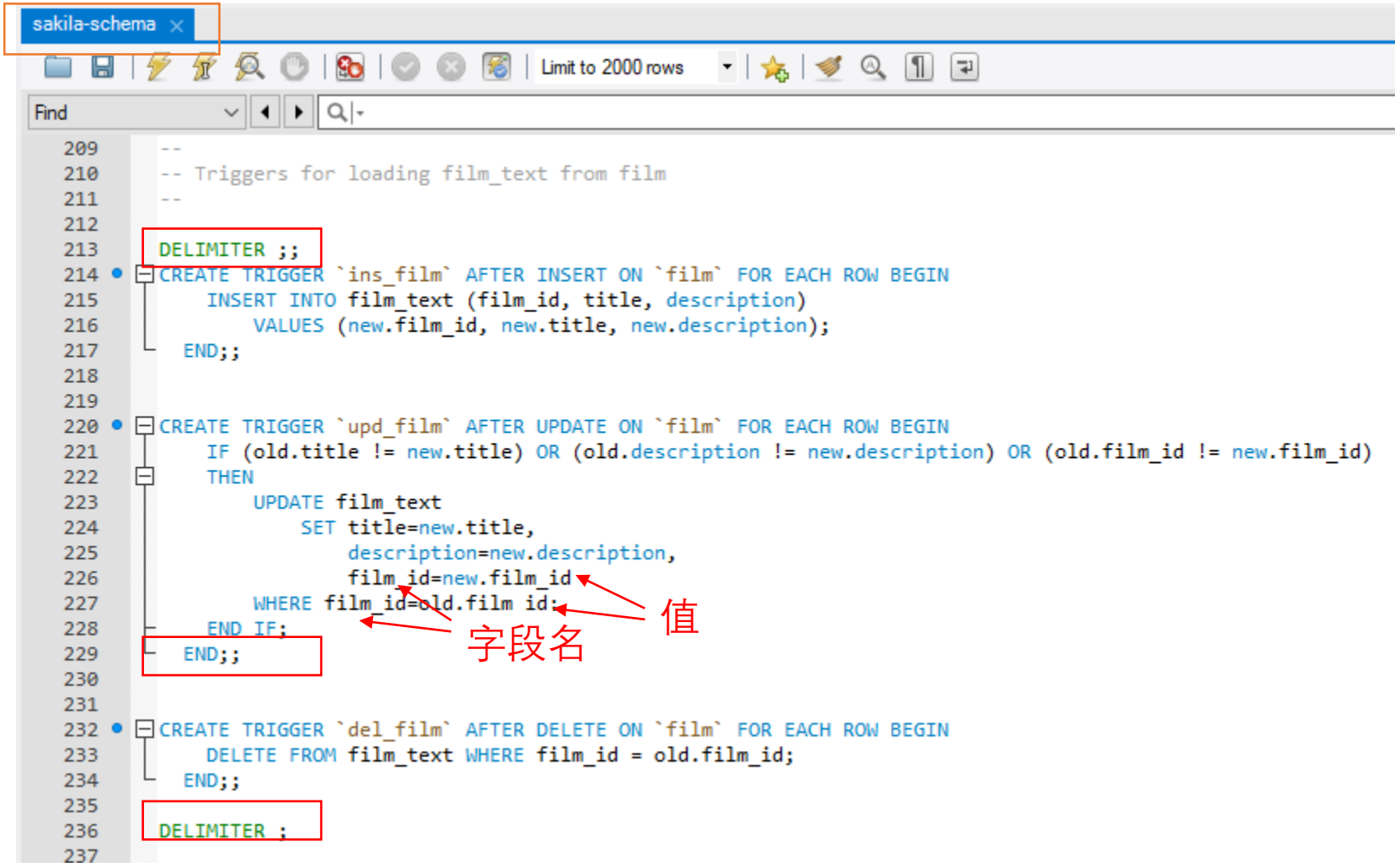
实验原理

触发器（Triggers）



实验原理

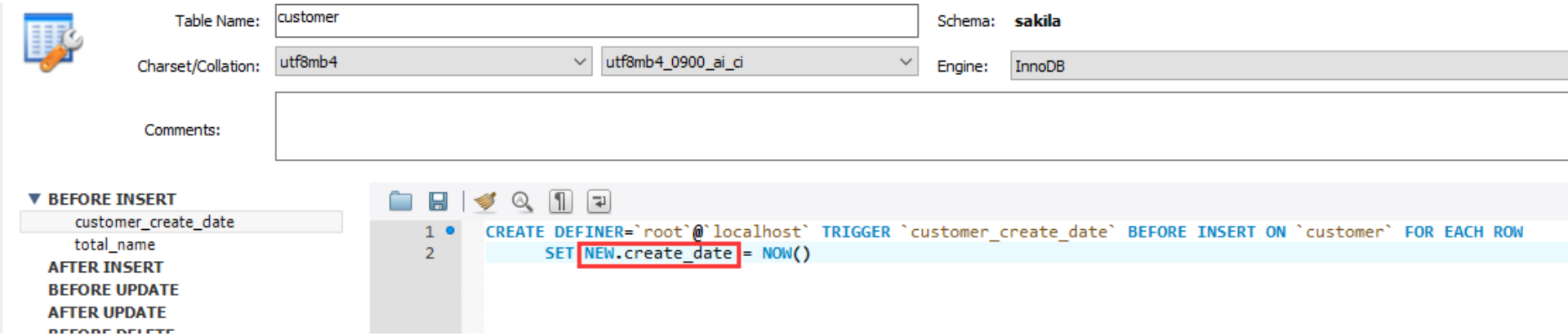
触发器（Triggers）




```
209  --
210  -- Triggers for loading film_text from film
211  --
212
213  DELIMITER ;;
214  CREATE TRIGGER `ins_film` AFTER INSERT ON `film` FOR EACH ROW BEGIN
215      INSERT INTO film_text (film_id, title, description)
216          VALUES (new.film_id, new.title, new.description);
217    END;;
218
219
220  CREATE TRIGGER `upd_film` AFTER UPDATE ON `film` FOR EACH ROW BEGIN
221      IF (old.title != new.title) OR (old.description != new.description) OR (old.film_id != new.film_id)
222      THEN
223          UPDATE film_text
224              SET title=new.title,
225                  description=new.description,
226                  film_id=new.film_id
227          WHERE film_id=old.film_id;
228      END IF;
229    END;;
230
231
232  CREATE TRIGGER `del_film` AFTER DELETE ON `film` FOR EACH ROW BEGIN
233      DELETE FROM film_text WHERE film_id = old.film_id;
234    END;;
235
236  DELIMITER ;
237
```

实验原理

触发器（Triggers）



注意：new.字段的值可以在before类型的触发器中进行赋值和取值，在after类型的触发器中只能取值。

实验原理

约束（Constraint)

作用：为了确保表中的数据的正确性、有效性、完整性 。

常用约束：

- 主键：primary key
- 非空约束：not null
- 唯一约束：unique
- 外键约束：foreign key
- ……

实验原理

约束（Constraint）

```
CREATE TABLE rental (
  rental_id INT NOT NULL AUTO_INCREMENT,
  rental_date DATETIME NOT NULL,
  inventory_id MEDIUMINT UNSIGNED NOT NULL,
  customer_id SMALLINT UNSIGNED NOT NULL,
  return_date DATETIME DEFAULT NULL,
  staff_id TINYINT UNSIGNED NOT NULL,
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (rental_id),
  UNIQUE KEY  (rental_date,inventory_id,customer_id),
  KEY idx_fk_inventory_id (inventory_id),
  KEY idx_fk_customer_id (customer_id),
  KEY idx_fk_staff_id (staff_id),
  CONSTRAINT fk_rental_staff FOREIGN KEY (staff_id) REFERENCES staff (staff_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_rental_inventory FOREIGN KEY (inventory_id) REFERENCES inventory (inventory_id) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_rental_customer FOREIGN KEY (customer_id) REFERENCES customer (customer_id) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

实验原理

存储过程（Procedure）& 函数（Function）

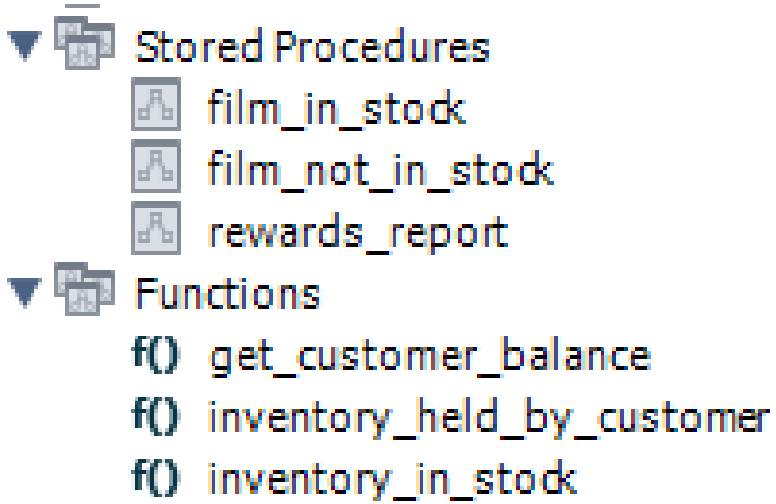
- 可统称为存储程序
- 是一组为了完成特定功能的SQL语句集

- 优点：
  ① 复用：存储的程序可重用
  ② 安全：仅授予某些用户访问存储程序的权限，而不需要提供访问基础数据库表的任何权限

实验原理

存储过程（Procedure）& 函数（Function）

- sakila已有



实验原理

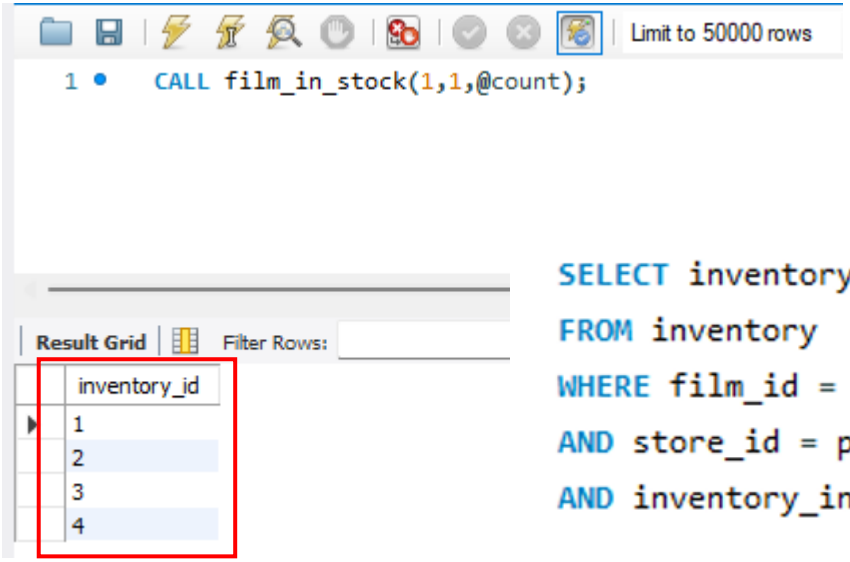
存储过程（Procedure）

```
587  DELIMITER $$
588
589  CREATE PROCEDURE film_in_stock(IN p_film_id INT, IN p_store_id INT, OUT p_film_count INT)
590  READS SQL DATA
591  BEGIN
592      SELECT inventory_id
593      FROM inventory
594      WHERE film_id = p_film_id
595      AND store_id = p_store_id
596      AND inventory_in_stock(inventory_id);
597
598      SELECT COUNT(*)
599      FROM inventory
600      WHERE film_id = p_film_id
601      AND store_id = p_store_id
602      AND inventory_in_stock(inventory_id)
603      INTO p_film_count;
604  END $$
605
606  DELIMITER ;
```

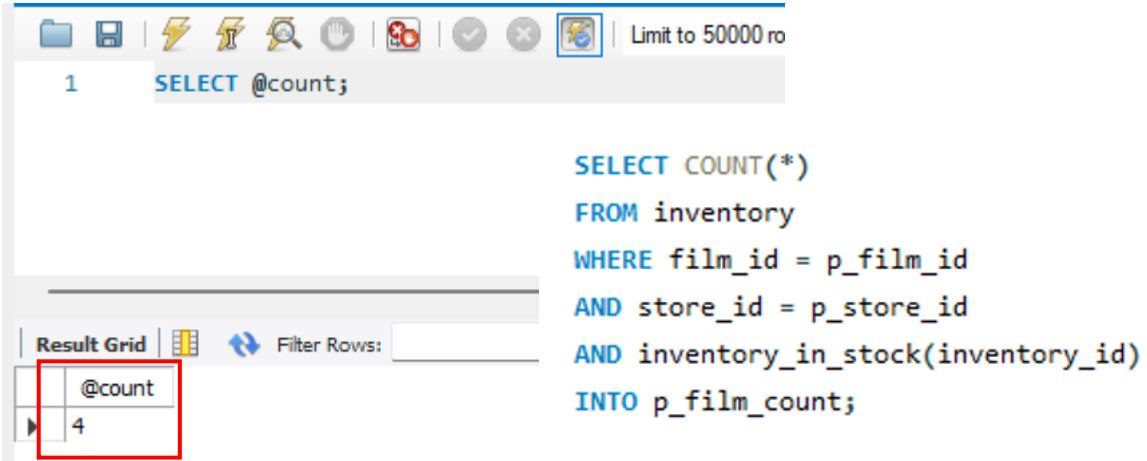
实验原理

存储过程（Procedure）

- 调用



- 获得返回值



实验原理

函数（Function）

```
647  DELIMITER $$
648
649  CREATE FUNCTION inventory_in_stock(p_inventory_id INT) RETURNS BOOLEAN
650  READS SQL DATA
651  BEGIN
652      DECLARE v_rentals INT;
653      DECLARE v_out     INT;
654
655      #AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE
656      #FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED
657
658      SELECT COUNT(*) INTO v_rentals
659      FROM rental
660      WHERE inventory_id = p_inventory_id;
661
662      IF v_rentals = 0 THEN
663        RETURN TRUE;
664      END IF;
665
666      SELECT COUNT(rental_id) INTO v_out
667      FROM inventory LEFT JOIN rental USING(inventory_id)
668      WHERE inventory.inventory_id = p_inventory_id
669      AND rental.return_date IS NULL;
670
671      IF v_out > 0 THEN
672        RETURN FALSE;
673      ELSE
674        RETURN TRUE;
675      END IF;
676  END $$
677  DELIMITER ;
```

实验原理

函数（Function）

```
647     DELIMITER $$
648
649 •   CREATE FUNCTION inventory_in_stock(p_inventory_id INT) RETURNS BOOLEAN
650     READS SQL DATA
651     BEGIN
          ……主要逻辑
676     END $$
677
678     DELIMITER ;
```

实验原理

函数（Function）

```
651  BEGIN
652      DECLARE v_rentals INT;
653      DECLARE v_out     INT;
654
655      #AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE
656      #FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED
657
658      SELECT COUNT(*) INTO v_rentals
659      FROM rental
660      WHERE inventory_id = p_inventory_id;
661
662      IF v_rentals = 0 THEN
663        RETURN TRUE;
664      END IF;
665
666      SELECT COUNT(rental_id) INTO v_out
667      FROM inventory LEFT JOIN rental USING(inventory_id)
668      WHERE inventory.inventory_id = p_inventory_id
669      AND rental.return_date IS NULL;
670
671      IF v_out > 0 THEN
672        RETURN FALSE;
673      ELSE
674        RETURN TRUE;
675      END IF;
676  END $$
```

该inventory_id从来没有出租过，返回true

该inventory_id借出去没有归还，返回false
否则，返回true

实验原理

函数（Function）

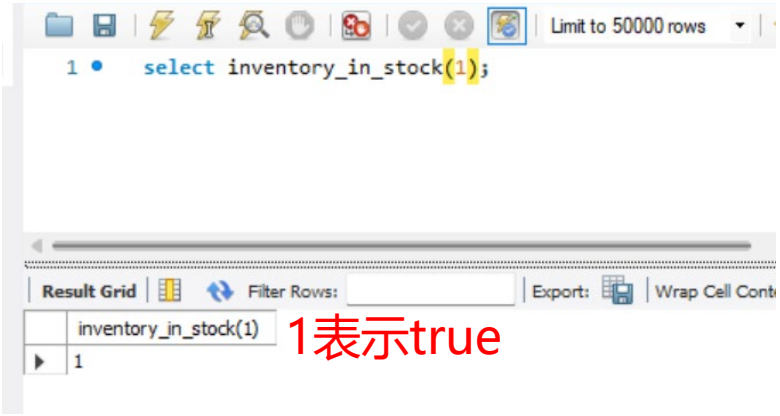
- 使用

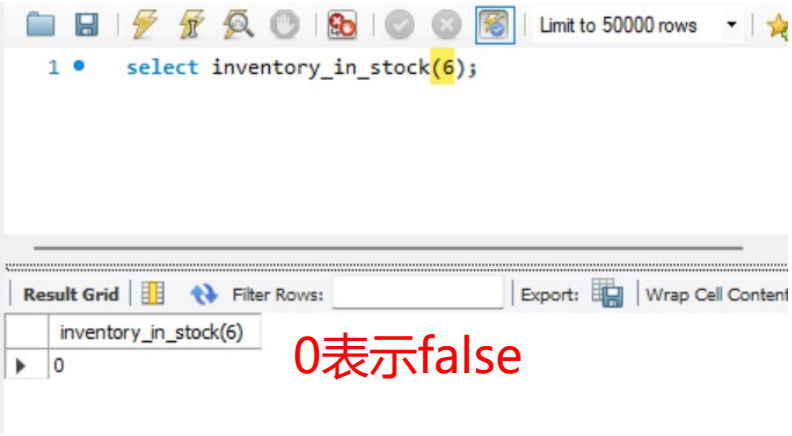
```
587     DELIMITER $$
588
589 ●   CREATE PROCEDURE film_in_stock(IN p_film_id INT, IN p_store_id INT, OUT p_film_count INT)
590     READS SQL DATA
591     BEGIN
592         SELECT inventory_id
593         FROM inventory
594         WHERE film_id = p_film_id
595         AND store_id = p_store_id
596         AND inventory_in_stock(inventory_id);
597
598         SELECT COUNT(*)
599         FROM inventory
600         WHERE film_id = p_film_id
601         AND store_id = p_store_id
602         AND inventory_in_stock(inventory_id)
603         INTO p_film_count;
604     END $$
605
606     DELIMITER ;
```

实验原理

函数（Function）

- 使用





实验原理

结构化查询语言(Structured Query Language, SQL)

- DDL - Data Definition Language，数据定义语言
- DML - Data Manipulation Language，数据处理语言
- **DCL - Data Control Language，数据控制语言**

  控制数据的访问权限，只有被授权的用户才能操作数据。

1. 创建用户

```
create user [用户名] identified by [登录密码];
```

2. 删除用户

```
drop user [用户名];
```

3. 用户授权

```
grant [权限1,权限2,…] on [数据库名].[表名] to [用户名];
```

4. 撤销授权

```
revoke [权限1,权限2,…] on [数据库名].[表名] from [用户名];
```

实验步骤

1. 跟随实验指导书的指引观察和分析Sakila数据库中的视图、触发器、约束、存储过程和函数，并回答实验指导书中的问题。

2. 根据应用场景，为Sakila数据库合理地设计并实现：

- 1个视图
- 1个触发器
- 1个存储过程

要求：视图需关联至少2个表，触发器需验证是否生效，存储过程需调用请在报告中提供创建语句和调用结果截图。

3. 根据实验指导书要求，创建新用户并为之分配权限。

4. 【附加题】思考问题：在阿里开发规范里有一条“**【强制】不得使用外键与级联，一切外键概念必须在应用层解决。**”请分析一下原因。外键是否没有存在的必要？

作业提交

- 课后提交：提交实验报告至作业提交平台（截止日期参考平台发布）

作业平台入口：http://grader.tery.top:8000/#/login
用户名、密码默认是你的学号

- 推荐使用 Chrome 浏览器
- 注意提交 pdf 格式的报告

同学们
请开始实验吧！