**深 圳 大 学**

**实 验 报 告**

**课程名称： 数据库系统**

**实验序号： 实验3**

**实验名称：**  **DMBS应用**

**学 号： 2022155028**

**姓 名： 黄亮铭**

**实验完成日期： 2024年 11 月 06 日**

观察并回答问题

关于视图-回答问题：

（1）分析以下3 个视图，回答以下问题：

|  |
| --- |
| SELECT DBMS\_METADATA.GET\_DDL('VIEW', 'ACTOR\_INFO','SAKILA') DDL FROM  DUAL;  SELECT DBMS\_METADATA.GET\_DDL('VIEW', 'FILM\_LIST','SAKILA') DDL FROM  DUAL;  SELECT DBMS\_METADATA.GET\_DDL('VIEW',  'SALES\_BY\_FILM\_CATEGORY','SAKILA') DDL FROM DUAL; |

|  |  |  |
| --- | --- | --- |
| 视图名 | 关联表 | 作用 |
| actor\_info | ACTOR | 快速访问演员信息 |
| film\_list | FILM | 方便展示电影信息 |
| sales\_by\_film\_category | FILM\_CATEGORY | 便于分析电影票房 |

（2） 创建视图分别执行以下3 句SQL 语句：

创建视图：

|  |
| --- |
| CREATE OR REPLACE VIEW sakila.customer\_simple\_view AS  SELECT customer\_id,  first\_name,  last\_name,  email,  address\_id,  active,  create\_date,  last\_update  FROM customer; |

执行SQL语句：

|  |
| --- |
| (1)update sakila.customer\_simple\_view set email =  'newemail@example.com' where customer\_id = 1;  (2)update sakila.staff\_list set "zip code" = '518055' where  ID = '1';  (3)update sakila.film\_list set price = 1.99 where FID = '1';  6 |

截图执行结果，并分析一下视图在什么情况下可以进行update 操作，什么

情况下不能？MYSQL和崖山数据库有什么区别

创建视图的命令执行结果如下图所示。

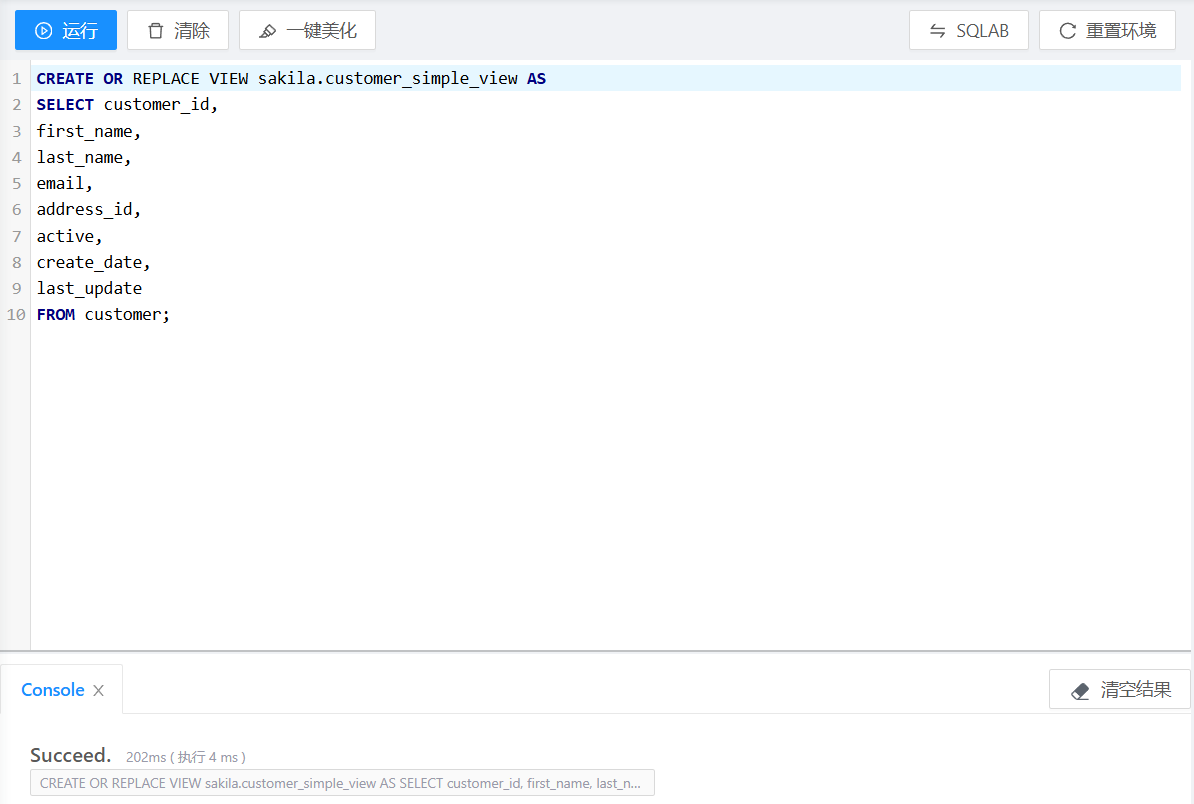


图1：创建视图

三条SQL语句执行结果如图2所示。由执行结果可以得出，只有第一条SQL语句执行成功，第二条和第三条SQL语句执行失败，发生了不同的错误。

第二条SQL语句错误的原因是尝试更新了非键保持的表；第三条SQL语句错误的原因是视图上的数据操作不合法。这两种错误的可能原因为：1）视图包含了聚合函数、group by子句、distinct关键字和子查询等；2）视图没有直接映射到一个具体的表，并且更新操作影响了多个表。

通过上述分析，我认为视图在满足下列条件的前提下，可以进行update操作。如果不满足下列条件，则不能对视图进行update操作。需要满足的条件：1）创建的视图不包含了聚合函数、group by子句、distinct关键字和子查询等；2）视图是基于多个表生成时，一次update操作只能影响一个表；3）创建视图时，select语句选择的列没有派生列。

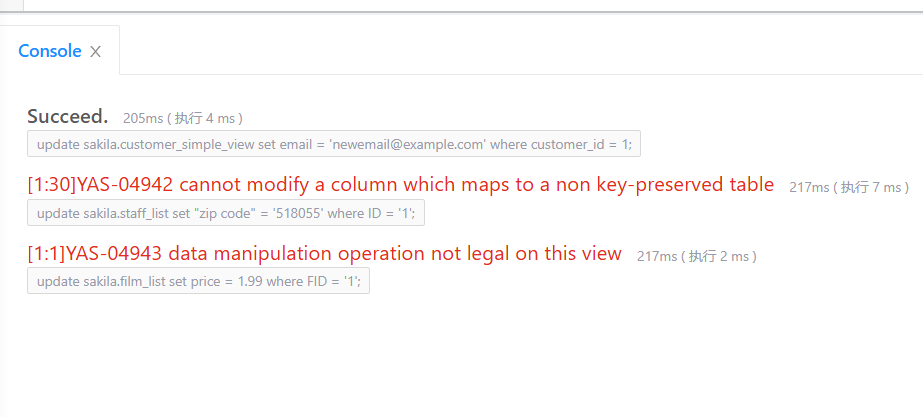


图2：SQL执行结果

关于触发器-回答问题：

1. 触发器rental\_date建在哪个表上？这个触发器实现什么功能？在这个表上新增一条数据，验证一下触发器是否生效。（截图语句和执行结果）

触发器rental\_date建立在表RENTAL上。这个触发器实现了如下功能：当有新记录插入表中时，触发器会自动将新记录的RENTAL\_DATE字段设置为当前系统的日期和时间。

执行下表格中的SQL语句。

|  |
| --- |
| INSERT INTO  SAKILA.RENTAL (  RENTAL\_ID,  RENTAL\_DATE,  INVENTORY\_ID,  CUSTOMER\_ID,  RETURN\_DATE,  STAFF\_ID,  LAST\_UPDATE  )  VALUES  (  432802,  '2024-10-20 00:02:21',  1824,  399,  NULL,  1,  '2024-10-21 21:30:53'  ); |

执行结果如下图所示。



图3：SQL执行结果

然后，我们通过RENTAL\_ID字段查找上面插入的数据，如果触发器生效，则RENTAL\_DATE字段会被修改为‘2024-10-30 HH:MM:SS’。如果触发器不生效，则RENTAL\_DATE字段不会被修改。

执行如下表的查询语句。

|  |
| --- |
| SELECT \* FROM SAKILA.RENTAL WHERE RENTAL\_ID = 432802; |

执行结果如下图所示。发现RENTAL\_DATE字段被修改，说明触发器生效了。

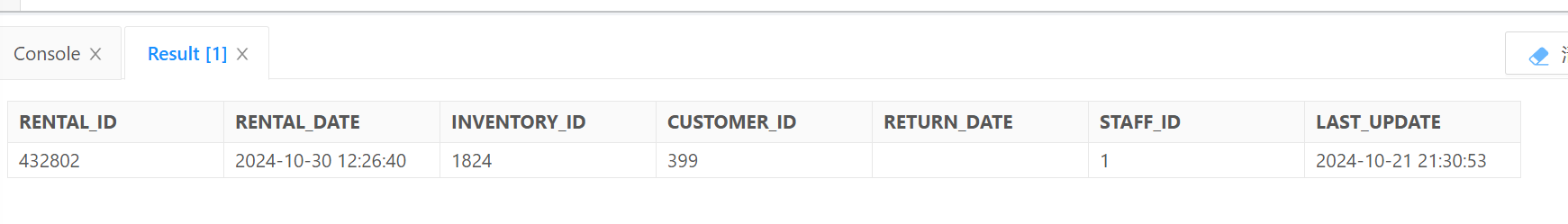


图4：查询结果

1. 触发器del\_film 建在哪个表上？这个触发器实现什么功能？在这个表上删除一条记录，验证一下触发器是否生效。（截图语句和执行结果）

触发器del\_film建立在表FILM上。这个触发器实现如下功能：当表FILM删除一条记录时，表FILE\_TEXT会同步删除FILE\_TEXT.FILE\_ID = OLD.FILE\_ID的记录。

首先执行如下插入语句和查询语句。

|  |
| --- |
| INSERT INTO SAKILA.FILM  (FILM\_ID, TITLE, DESCRIPTION, RELEASE\_YEAR, LANGUAGE\_ID, ORIGINAL\_LANGUAGE\_ID, RENTAL\_DURATION, RENTAL\_RATE, LENGTH, REPLACEMENT\_COST, RATING, SPECIAL\_FEATURES, LAST\_UPDATE)  VALUES  (1001, 'TEST\_FILM', 'THIS IS A TEST, NOT A FILM', 2024, 1, NULL, 1, 9.99, 120, 29.99, NULL, NULL, SYSDATE);  SELECT \* FROM SAKILA.FILM WHERE FILM\_ID = 1001;  SELECT \* FROM SAKILA.FILM\_TEXT WHERE FILM\_ID = 1001; |

查询结果如下图所示，说明数据被成功插入，FILM和FILM\_TEXT中均存在FILM\_ID=1001的数据记录，其中FILM\_TEXT中的记录为触发器自动插入。

|  |
| --- |
|  |
| 图5a：FILM表查询结果 |
|  |
| 图5b：FILM\_TEXT表查询结果 |

执行删除FILM\_ID为1001的数据记录的SQL语句，执行结果如下图所示，说明FILM中FILM\_ID=1001的数据记录被成功地删除。

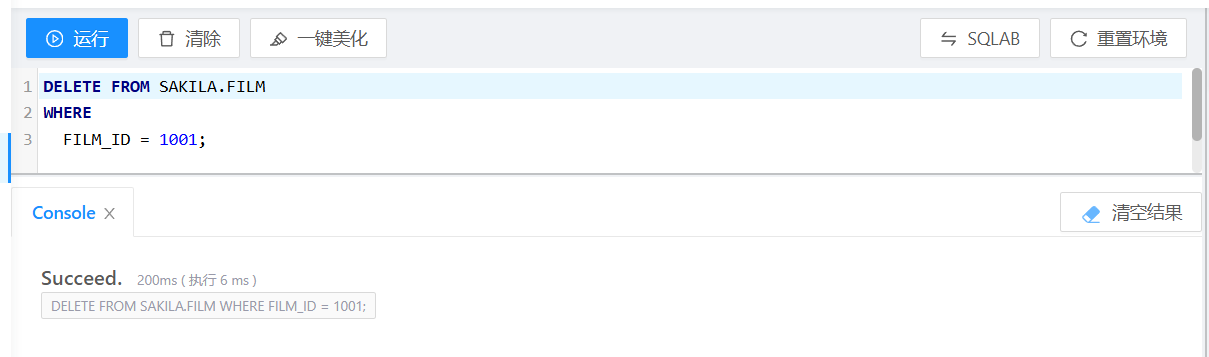


图6：执行结果

执行如下的查询语句，执行语句成功，但是查询结果为空。说明相关记录已经被删除，进而证明触发器生效。

|  |
| --- |
|  |
| 图7a：执行SQL语句结果 |
|  |
| 图7b：FILM表查询结果 |
|  |
| 图7c：FILM\_TEXT表查询结果 |

1. 现在有这样一个建立触发器的语句，如下图所示。此时需要迁移payment表历史记录，请同学们思考，这个触发器是否会生效？如果生效的话，有什么办法可以避免迁移记录的payment\_date被修改？



分析:

触发器PAYMENT\_DATE设置为在PAYMENT表上执行INSERT操作之前触发，并且会将新插入行的payment\_date字段设置为系统当前时间戳。

在迁移payment表历史记录时会涉及到INSERT操作，因此，触发器会生效。触发器会自动地将新插入的数据记录的payment\_date字段设置为系统当前时间戳。

想要迁移记录的payment\_date不被修改，我们可以1）在迁移记录开始之前禁用触发器，在迁移记录结束之后重新开启触发器；2）使用数据库自带的导入导出工具。

关于约束-回答问题

现有rental表的建表语句如下：



1. rental表上建了哪几种约束？这些约束分别实现什么功能？

|  |  |
| --- | --- |
| 约束类型 | 功能 |
| NOT NULL，非空约束 | 字段不能为空，确保字段总是有效 |
| FOREIGN KEY，外键约束 | 确保引用的字段的值出现在另一个表中，维护参照完整性 |
| UNIQUE，唯一约束 | 确保某一列或某一组合的值是唯一的 |
| PRIMARY KEY， | 确保某一列或某一组合的值是唯一的，每一行记录都可以被其唯一标识 |

对于rental表来说，**1）NOT NULL**确保RENTAL\_ID、RENTAL\_DATE、INVENTORY\_ID、CUSTOMER\_ID、RETURN\_DATE和STAFF\_ID字段不为空；**2）FOREIGN KEY**确保rental表中的CUSTOMER\_ID列的值必须在SAKILA.CUSTOMER表的CUSTOMER\_ID列中存在，确保rental表中的INVENTORY\_ID列的值必须在SAKILA.INVENTORY表的INVENTORY\_ID列中存在，确保rental表中的STAFF\_ID列的值必须在SAKILA.STAFF表的STAFF\_ID列中存在；**3）UNIQUE**确保RENTAL\_DATE、INVENTORY\_ID和CUSTOMER\_ID这三个列的组合值在rental表上是唯一的；**4）PRIMARY KEY**确保RENTAL\_ID列的值是唯一的，并且每一行记录都可以被唯一标识。

1. 图中的ON DELETE RESTRICT 和ON UPDATE CASCADE 是什么意思？

ON DELETE RESTRICT 和ON UPDATE CASCADE是外键约束的行为选项，用于定义当被引用的数据（即外键指向的主键）被删除或更新时，外键如何处理。

ON DELETE RESTRICT：当尝试删除主键表中的记录时，如果存在外键表中的数据引用了这个主键，那么删除操作会被限制，即不允许删除。这确保了数据的参照完整性，防止出现不合法的引用记录。

ON UPDATE CASCADE：当主键表中的记录被更新时，如果存在外键表中的数据引用了这个主键，那么外键表中相应的记录也会自动更新，以保持数据的一致性。

关于存储过程-回答问题

1. 这个存储过程实现了什么功能？输出参数count\_rewardees 是什么？

从payment表中查询上个月内购买金额超过min\_dollar\_amount\_purchased且购买次数超过min\_monthly\_purchases的客户，将他们放到一个集合里，然后输出每个客户的详细信息。

输出参数count\_rewardees是满足上述要求的客户的数量。

1. last\_month\_start := ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), -1);具体是怎么获取到上个月第一天的日期的？请展开说明。

首先使用SYSDATE函数获取当前系统日期和时间，然后使用TRUNC(SYSDATE, 'MM')将当前系统日期和时间截断到月份的第一天，即去掉日和时间部分，只保留年和月，最后使用ADD\_MONTHS(TRUNC(SYSDATE, 'MM'), -1)函数将当前系统日期的月份减1，即获取上个月的第一天。

关于函数-回答问题：

观察SAKILA用户里面的get\_customer\_balance 函数。

1. 这个函数实现了什么功能？返回值是什么？

这个函数的功能是计算给定客户ID和特定生效日期下的当前余额，影响的因素有租赁费用、逾期费用、超期租赁费用和已支付的款项。

返回值是一个NUMBER类型的数值，代表计算出的当前余额。

1. 这个函数体中用到了3 个函数，是哪几个函数？这3 个函数的作用分别是？

|  |  |
| --- | --- |
| 函数 | 作用 |
| NVL(a,b) | NVL函数用于替换NULL值为指定的值。如果a为NULL，则NVL函数将其替换为b |
| SUM(a) | 计算a列的总和 |
| TRUNC(a) | 截断日期，去掉日期中的时间部分，只保留到日 |

创建新用户并授予权限

创建新用户

1. 查看当前已有用户：select username from dba\_users;

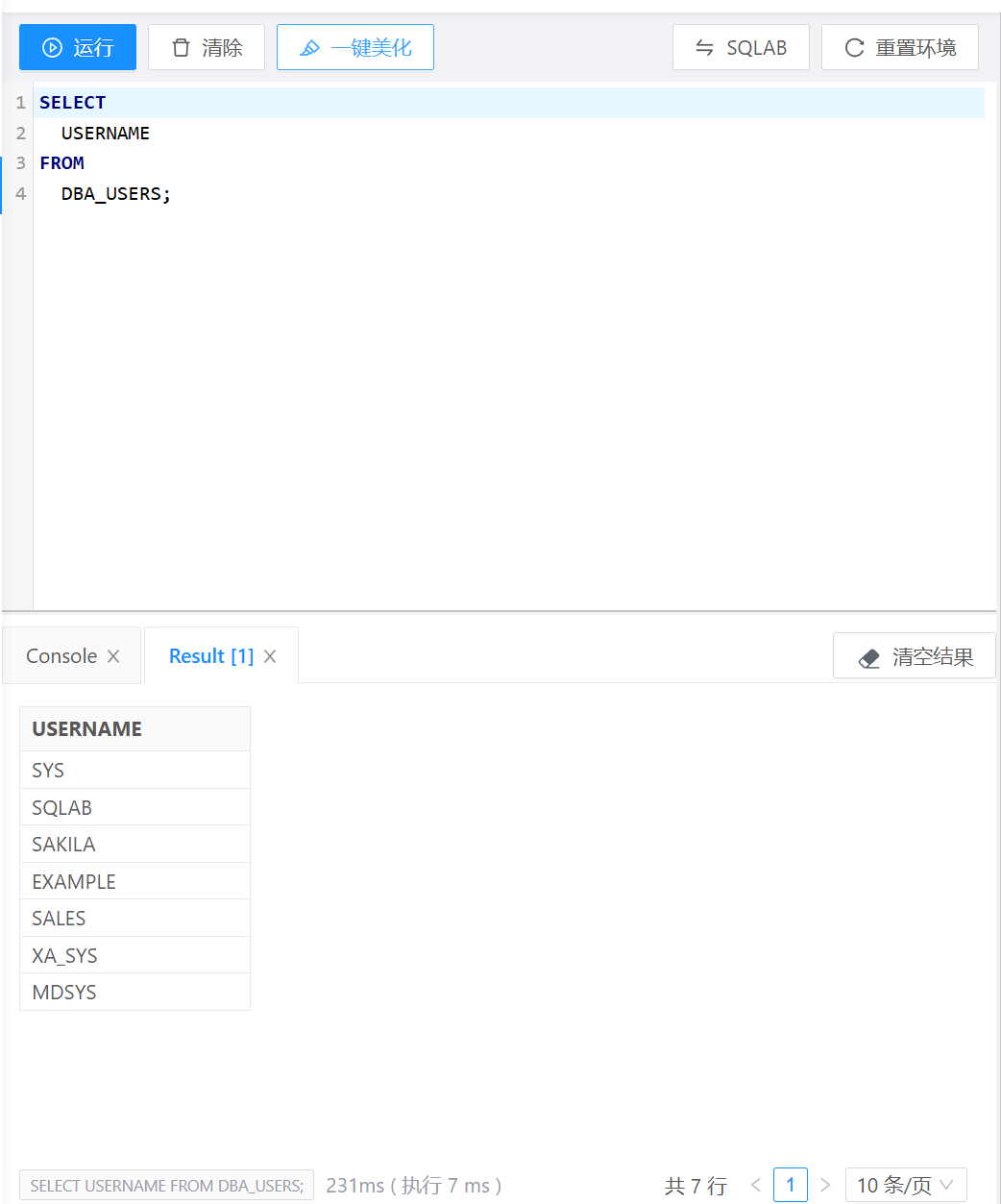


图8：查看用户

1. 新建SAKILA\_TESE用户（密码123456）：create user SAKILA\_TEST identified by 123456;执行结果如下图所示，说明新建用户成功。

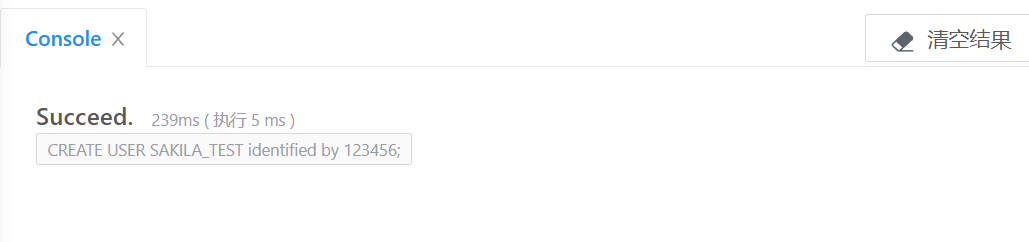


图9：语句执行结果

1. 再次查看已有用户：select username from dba\_users;新用户已经被添加到列表中。

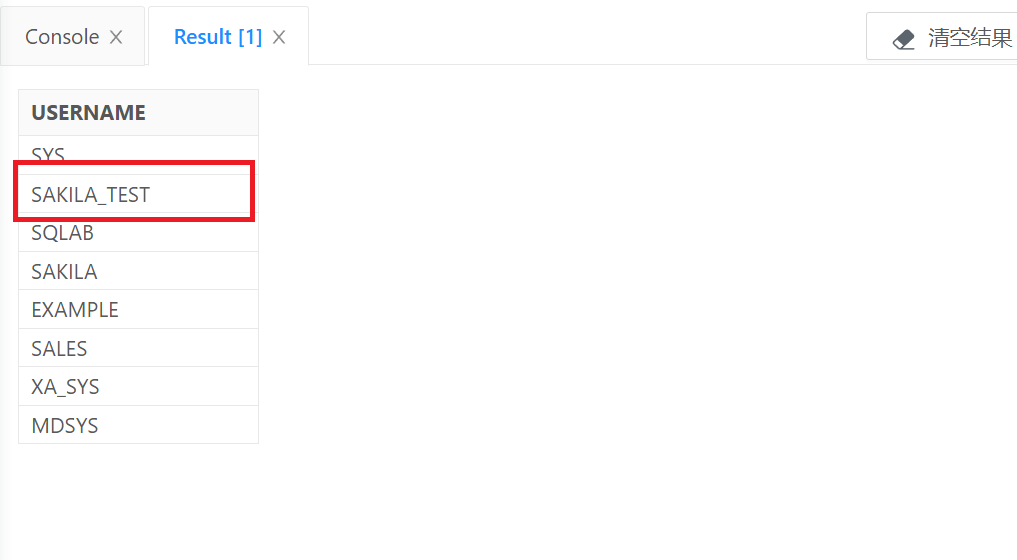


图10：新建用户结果

为新用户SAKILA\_TEST赋予访问sakila 的权限

1. 查看新用户当前的权限。

|  |
| --- |
| select \* from dba\_sys\_privs where grantee='SAKILA\_TEST';  select \* from dba\_tab\_privs where grantee='SAKILA\_TEST';  select \* from dba\_role\_privs where grantee='SAKILA\_TEST'; |

执行结果如下图所示，发现新用户没有任何权限。

|  |
| --- |
|  |
|  |
|  |

图11：权限查询结果

1. 赋予新用户connect角色（CONNECT角色具有CREATE SESSION权限，通过赋予CONNECT角色使用户能够登录会话。）：grant connect to SAKILA\_TEST;赋予新用户resource角色（RESOURCE角色具有CREATE TABLE、CREATE SEQUENCE、CREATE PROCEDURE、CREATE TRIGGER、CREATE TYPE的权限。）：grant resource to SAKILA\_TEST;



图12：赋予权限的语句执行结果

1. 使用新用户连接数据库。



图13：新用户登录

1. 把原来的SAKILA用户中所有的表的访问权限赋予SAKILA\_TEST 用户：执行语句SELECT 'GRANT SELECT ON SAKILA.' || TABLE\_NAME || ' TO SAKILA\_TEST;' FROM ALL\_TABLES WHERE OWNER = 'SAKILA';然后将得到的赋予权限的语句输入到命令行。

执行结果如下图所示，说明赋予SAKILA\_TEST用户权限成功。

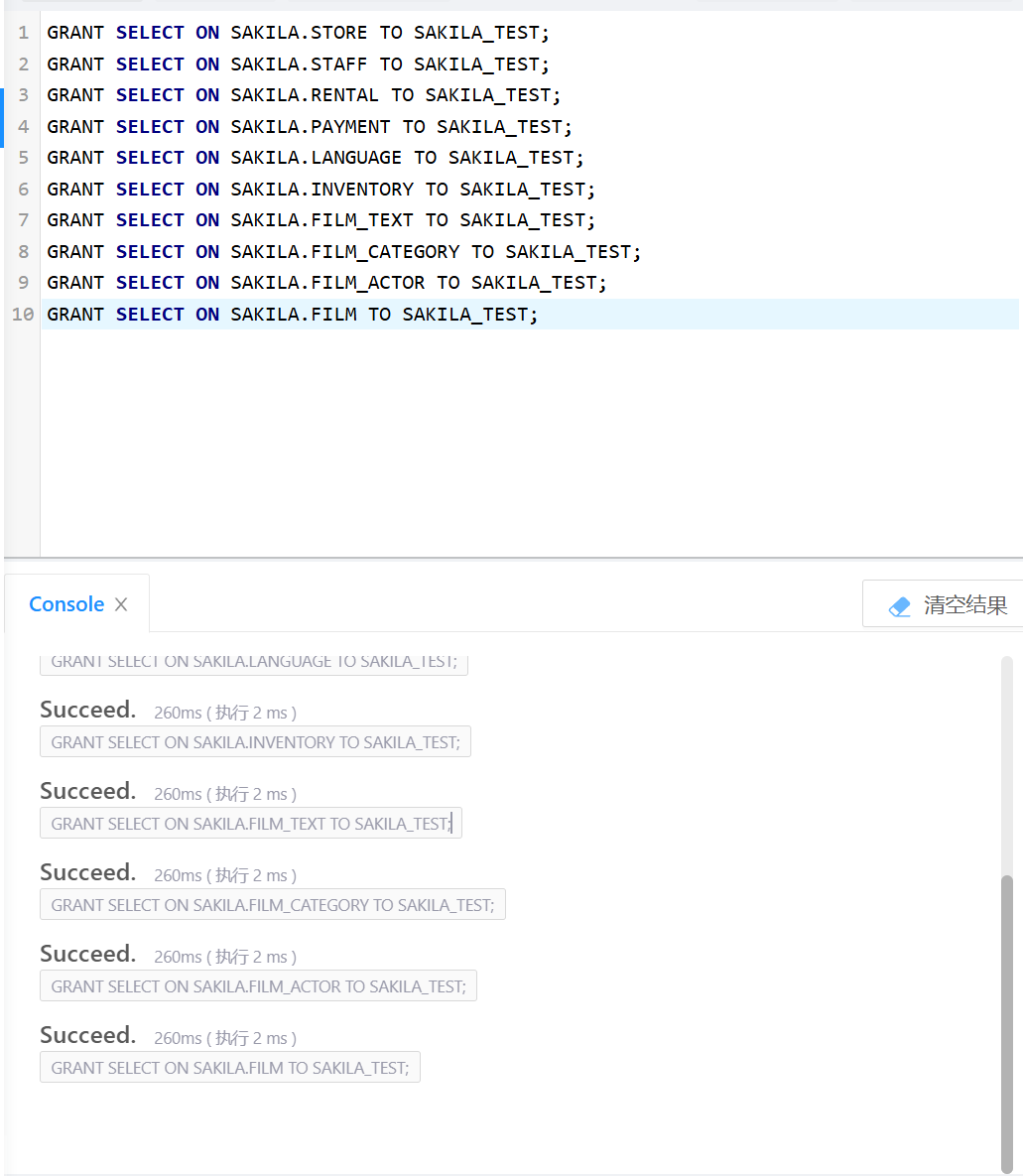


图14：赋予权限结果

1. 切换到sakila\_test 用户，这时可以看到sakila 用户的所有表，可以进行查询操作：SELECT \* FROM ALL\_TABLES;

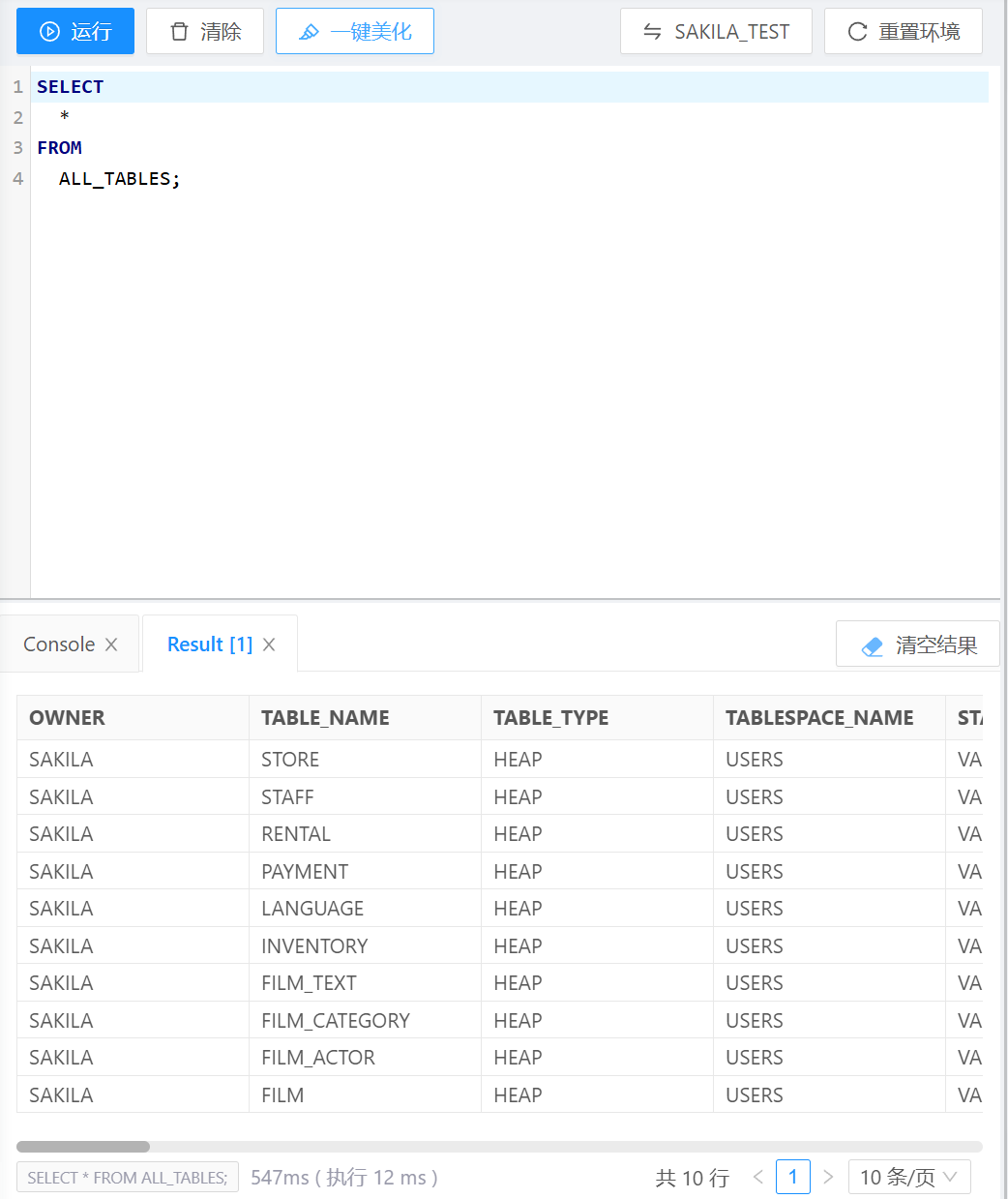


图15：查询结果

设计并实现

根据应用场景，为Sakila 数据库合理地设计并实现：

1. 设计1 个视图，通过关联customer和address 2 个表，可以查看客户及其地址信息的列表，方便用户查看地址信息；

设计思路：查看customer表，一共有8个字段。其中，customer\_id, first\_name, last\_name, email和address\_id是需要通过视图展示出来的客户信息。同时，我们还需要根据address\_id找到对应的address表中的信息。查看address表，其中需要我们关注的字段为ADDRESS, ADDRESSS2, DISTRICT, CITY\_ID, POSTAL\_CODE和PHONE。此时，我们还可以根据CITY\_ID去city表中找到对应的城市名，但是题目要求我们只需要关联customer和address2个表。

综合上述分析，我们只需要创建视图显示上面提到的字段即可。创建视图语句如图16所示。

|  |
| --- |
| CREATE VIEW sakila.customer\_address\_view AS  SELECT c.customer\_id, c.first\_name, c.last\_name, c.email, c.address\_id, a.address, a.address2, a.district, a.city\_id, a.postal\_code, a.phone  FROM sakila.customer AS c  JOIN sakila.address AS a ON c.address\_id = a.address\_id; |

图16：创建视图

创建视图语句运行结果如下图所示。终端提示Succeed说明创建视图语句成功运行。

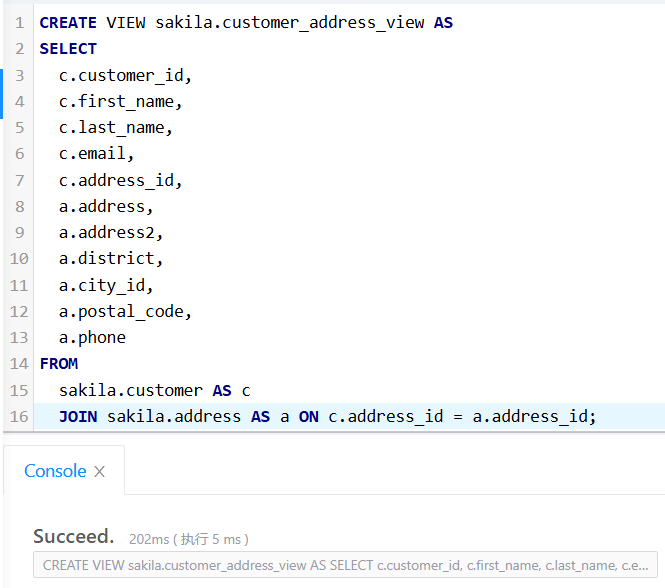


图17：创建语句运行结果

接下来我们通过视图查阅客户及其地址信息，查看上述语句创建的视图是否能正确显示客户及其地址信息。

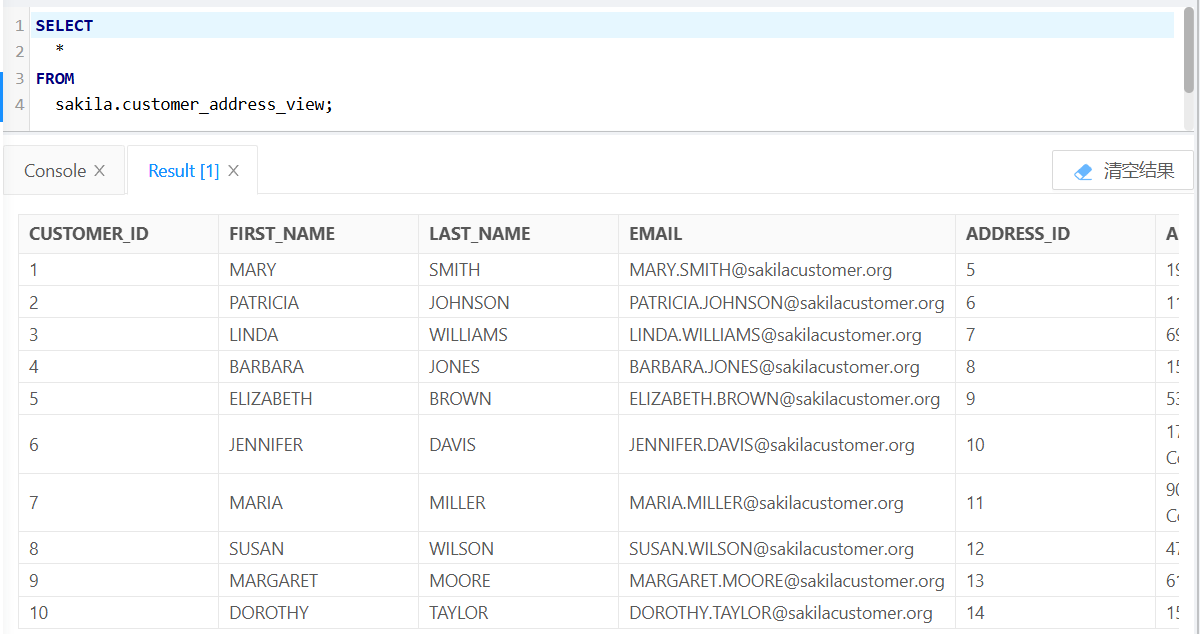


图18：显示客户及其地址信息

由图18可以看出，视图可以正确显示客户及其地址信息。

1. 设计1 个触发器，融合ins\_film、upd\_film和del\_film的功能，需要在报告里体现触发器生效；

我们首先将三个触发器实现的功能写到一个新的触发器中，然后修改触发状态，根据不同的触发状态进入不同的选择分支中。通过上述步骤，实现一个触发器融合三个触发器的功能。

|  |
| --- |
| CREATE OR REPLACE TRIGGER sakila.film\_sync\_trigger  BEFORE INSERT OR UPDATE OR DELETE ON sakila.film  FOR EACH ROW  BEGIN  IF INSERTING THEN  INSERT INTO sakila.film\_text (film\_id, title, description)  VALUES (:NEW.film\_id, :NEW.title, :NEW.description);  END IF;  IF UPDATING THEN  IF :OLD.title != :NEW.title OR :OLD.description != :NEW.description OR :OLD.film\_id != :NEW.film\_id THEN  UPDATE sakila.film\_text  SET title = :NEW.title,  description = :NEW.description,  film\_id = :NEW.film\_id  WHERE film\_id = :OLD.film\_id;  END IF;  END IF;  IF DELETING THEN  DELETE FROM sakila.film\_text  WHERE film\_id = :OLD.film\_id;  END IF;  END; |

图19：创建触发器

上述创建触发器的语句运行结果如下图。由下图终端显示的Succeed可知，触发器创建成功。

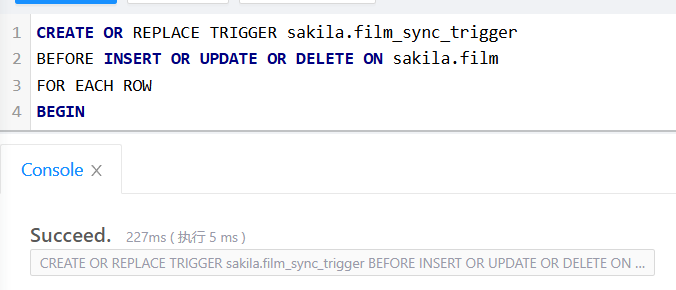


图20：创建结果

接下来我们需要知道触发器是否生效。查阅崖山数据库的手册得知，我们需要输入命令：。然后找到字段，如果字段的值为，则说明触发器已经生效，否则没有生效。

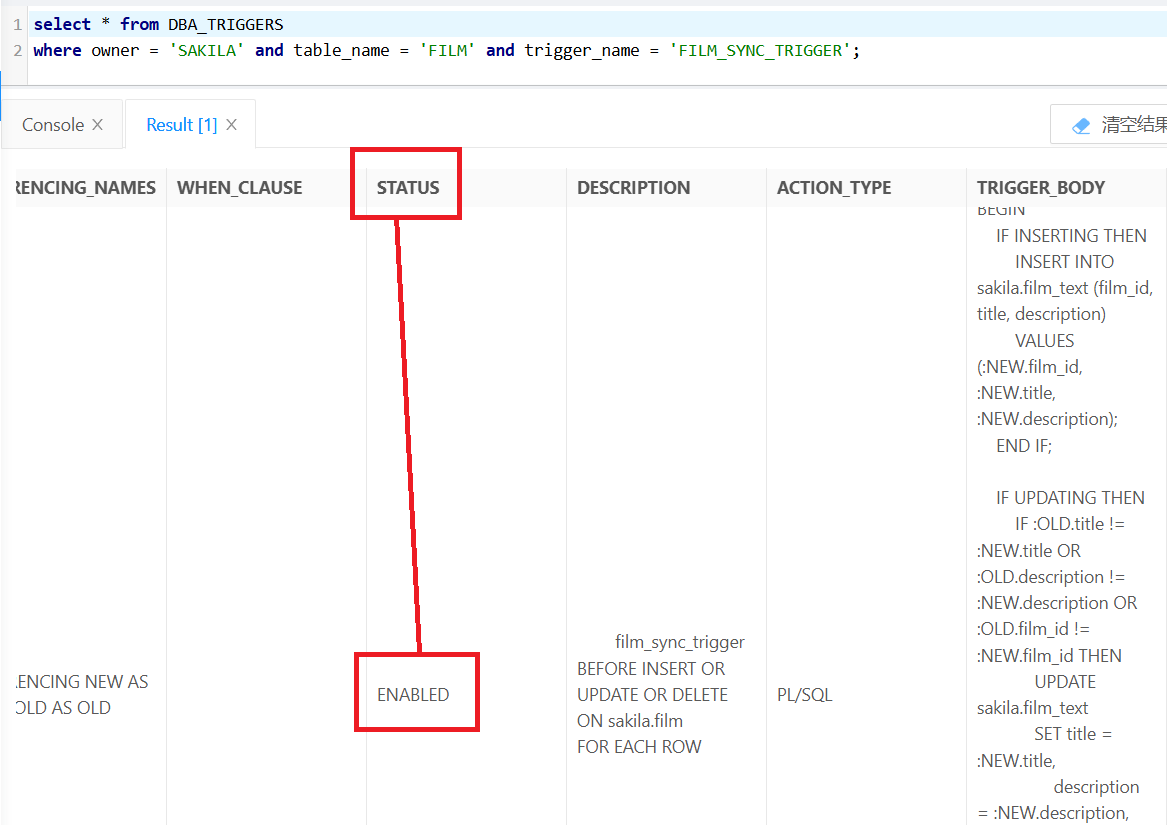


图21：查看触发器是否生效

由上图得知，触发器已经生效，其字段的值已经是。

1. 设计1 个存储过程，用于获取特定客户在指定时间段内的租赁历史记录。它将返回客户的租赁信息，包括租赁日期、归还日期、租赁的电影标题以及租赁费用。须在报告里调用，并展示结果。

设计的存储过程如下所示。1）参数为：客户的id、查询的起始时间和查询的结束时间；2）定义相关变量，用于临时存储一条租赁历史记录；3）定义一个游标用于遍历所有符合记录的租赁历史；4）使用游标和循环结构遍历所有符合条件的记录并展示。

|  |
| --- |
| DELIMITER $$  CREATE PROCEDURE get\_customer\_rental\_history(  IN p\_customer\_id INT,  IN p\_start\_date TIMESTAMP,  IN p\_end\_date TIMESTAMP  )  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE v\_rental\_date TIMESTAMP;  DECLARE v\_return\_date TIMESTAMP;  DECLARE v\_film\_title VARCHAR(128);  DECLARE v\_rental\_fee DECIMAL(4, 2);  DECLARE rental\_cursor CURSOR FOR  SELECT r.rental\_date, r.return\_date, f.title, f.rental\_rate  FROM rental AS r  JOIN inventory AS i ON r.inventory\_id = i.inventory\_id  JOIN film AS f ON i.film\_id = f.film\_id  WHERE r.customer\_id = p\_customer\_id  AND r.rental\_date BETWEEN p\_start\_date AND p\_end\_date;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  OPEN rental\_cursor;  read\_loop: LOOP  FETCH rental\_cursor INTO v\_rental\_date, v\_return\_date, v\_film\_title, v\_rental\_fee;  IF done THEN  LEAVE read\_loop;  END IF;  SELECT v\_rental\_date AS 'Rental Date',  v\_return\_date AS 'Return Date',  v\_film\_title AS 'Film Title',  v\_rental\_fee AS 'Rental Fee';  END LOOP;  CLOSE rental\_cursor;  END $$  DELIMITER ; |

执行结果如下图所示，红框中的OK说明创建存储过程成功。

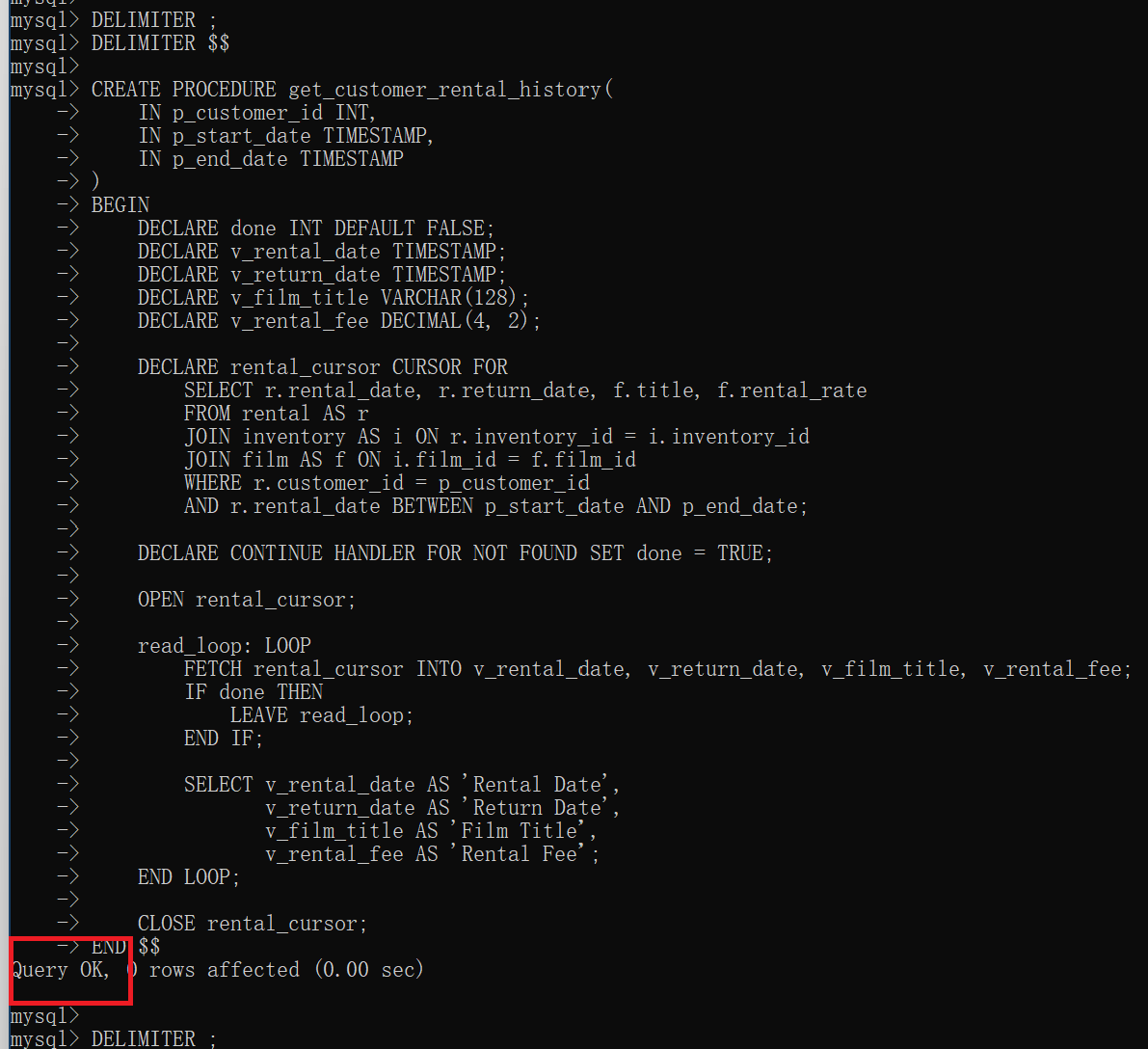


图22：创建存储过程

以客户1为例，首先查看客户1的所有租赁记录（按照租赁日期的升序排序）：

我们的目标是调用存储过程，查找客户1在2005年6月的租赁记录。因此，下图只展示属于我们的目标日期的部分。可以看到，符合要求的一共右7条记录。

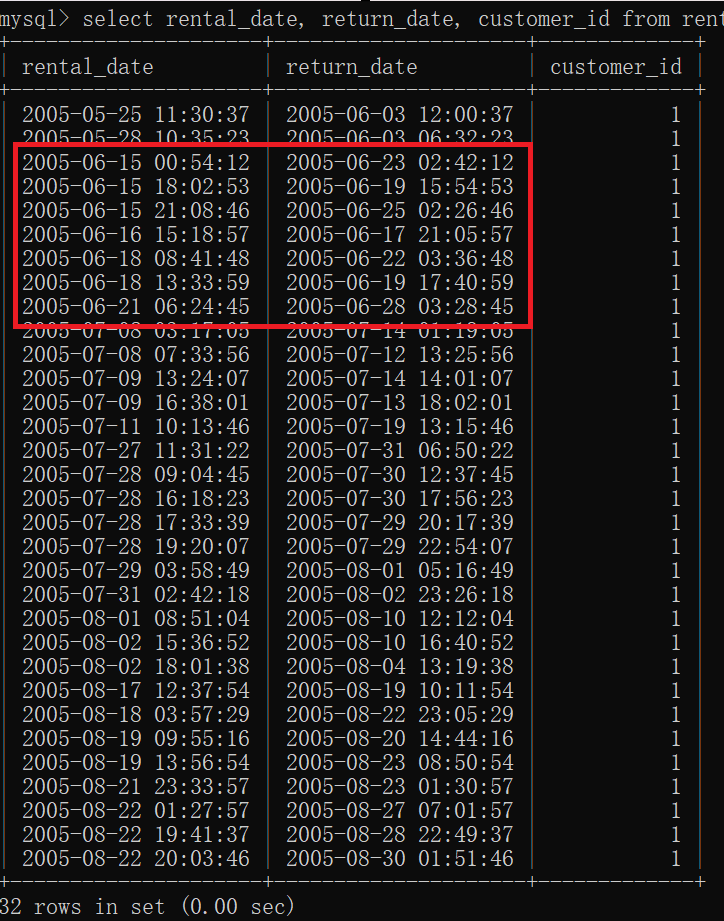


图23：客户1的租赁记录（部分）

调用上述存储过程：。结果如下图所示。一共有7条记录，并且记录的租赁日期和归还日期与上相符，说明我设计的存储过程的正确性。

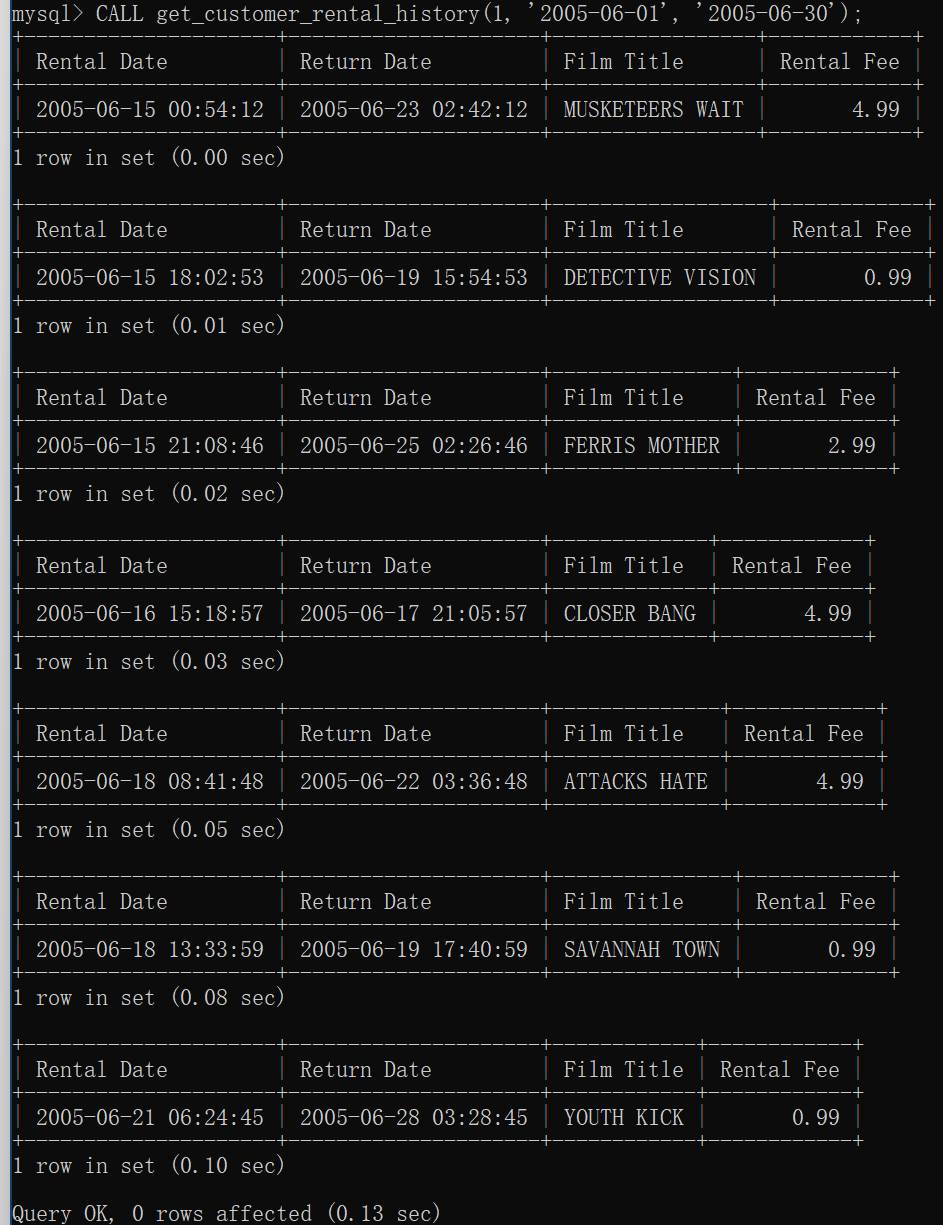


图24：调用存储过程查询结果

思考题

在MySQL开发规范中，通常要求为每张表创建一个主键，但是在YashanDB中，表

可以没有主键。请分析一下原因。主键是否没有存在的必要？

答：主键可以不存在的原因有：1）数据记录的字段包含足够的信息来唯一标识记录或者数据记录的字段可以组合成候选码；2）可能对于一些数据记录来说，并不需要保证唯一性；3）为了避免索引的开销。

但是，主键仍然有存在的必要。原因如下：1）使用主键可以保证数据的唯一性，没有重复的记录；2）在关系数据库中，其他表可以通过外键与主键相关联，从而维护数据的参照完整性；3）主键通常会自动创建索引，可以提高查询效率；4）主键是实体完整性的一部分，有助于维护数据的准确性和一致性。