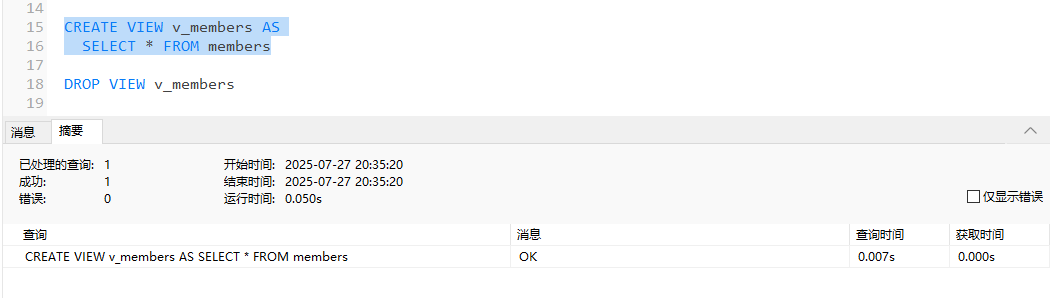
1.创建视图

CREATE VIEW v\_members AS

SELECT \* FROM members



2.删除视图

DROP VIEW v\_members



3.创建一个带参数的存储过程

CREATE PROCEDURE pro\_insert(

IN uname VARCHAR(50),

IN email VARCHAR(50))

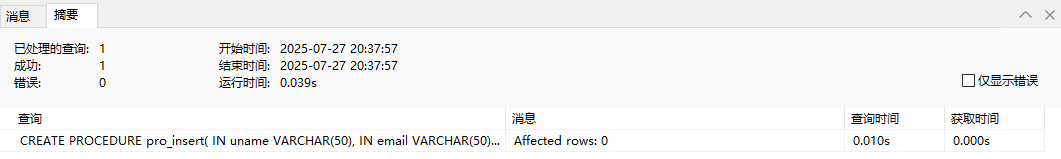
BEGIN

INSERT INTO members(member\_id,member\_name,member\_email,password)

VALUES(NULL,uname,email,'123');

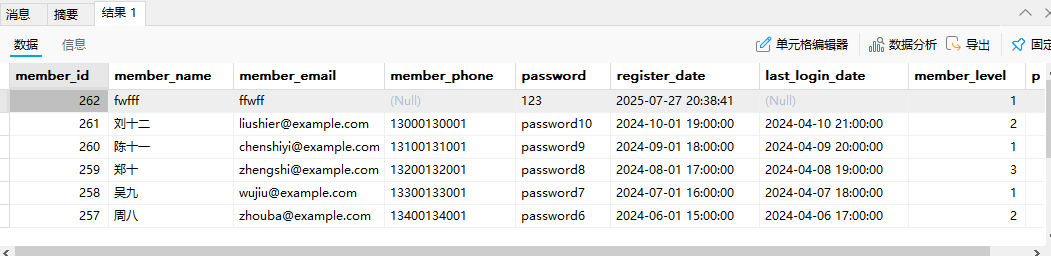
SELECT \* FROM members ORDER BY member\_id DESC;

END



4.调用存储过程

CALL pro\_insert('fwfff','ffwff')



5. 创建存储过程，WHILE循环

CREATE PROCEDURE pro\_insert\_10(

IN uname VARCHAR(50),

IN email VARCHAR(50))

BEGIN

DECLARE num INT DEFAULT 10;

DECLARE i INT DEFAULT 0;

WHILE i<num DO

INSERT INTO members(member\_id,member\_name,member\_email,password)

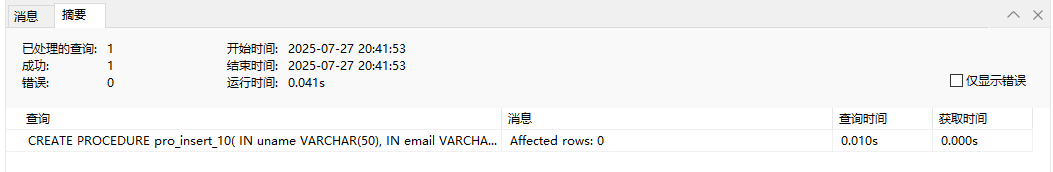
VALUES(NULL,CONCAT(uname,i),CONCAT(email,i),'123');

SET i = i+1;

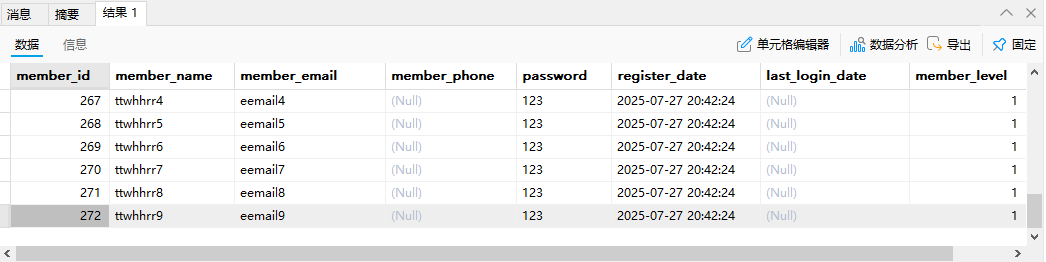
END WHILE;

SELECT \* FROM members WHERE 1;

END



6.调用存储过程，WHILE循环



7.触发器创建

CREATE TRIGGER tr\_updata

AFTER

INSERT ON orders

FOR EACH ROW

BEGIN

DECLARE pro\_num INT;

DECLARE pro\_unit\_price INT;

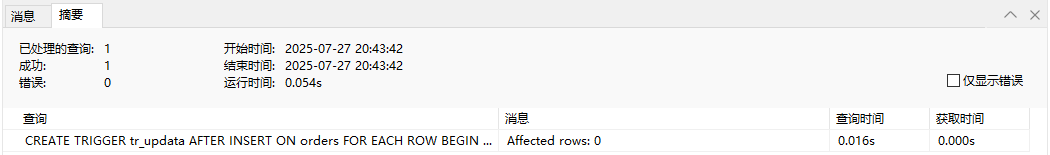
SELECT product\_id,price INTO pro\_num,pro\_unit\_price FROM products WHERE product\_id =3;

INSERT INTO order\_items(order\_id,product\_id,quantity,unit\_price)

VALUES (new.order\_id,pro\_num, pro\_unit\_price\*pro\_num , pro\_unit\_price);

UPDATE products SET stock = stock - 1;

END

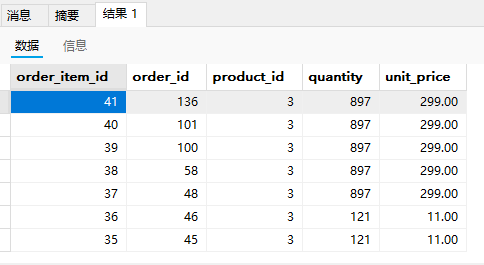


8.触发上面触发器

INSERT INTO orders(order\_id,member\_id,total\_amount,order\_status)

VALUES(NULL,'11',1500,'未支付');

9.查看是否触发



10.触发器中的NEW，OLD关键字

CREATE TRIGGER tr\_update\_order

AFTER UPDATE ON orders

FOR EACH ROW

BEGIN

IF OLD.order\_status='未支付' ANDNEW.order\_status='已支付' THEN

UPDATE products join order\_items join orders

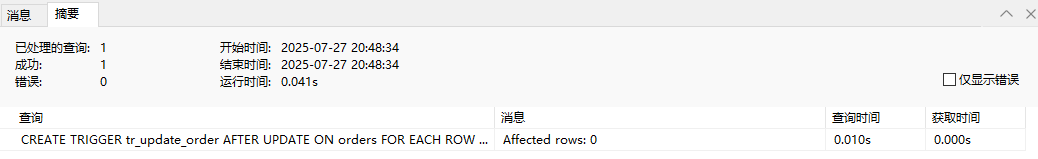
ON orders.order\_id=order\_items.order\_id and products.product\_id=order\_items.product\_id

SET stock = stock - quantity

WHERE orders.order\_id= NEW.order\_id;

END IF;

END;



11.触发器的三种触发方式

-- UPDATE 触发器

CREATE TRIGGER tr\_update\_order

AFTER UPDATE ON orders

FOR EACH ROW

BEGIN

END;

-- INSERT 触发器

CREATE TRIGGER tr\_insert\_order

AFTER INSERT ON orders

FOR EACH ROW

BEGIN

END;

-- DELETE 触发器

CREATE TRIGGER tr\_delete\_order

AFTER DELETE ON orders

FOR EACH ROW

BEGIN

END;

12. 新增订单，会员积分增加

CREATE TRIGGER tr\_insert\_orders

AFTER INSERT ON orders

FOR EACH ROW

BEGIN

UPDATE products join order\_items join orders

ON orders.order\_id=order\_items.order\_id and products.product\_id=order\_items.product\_id

SET stock = stock - quantity

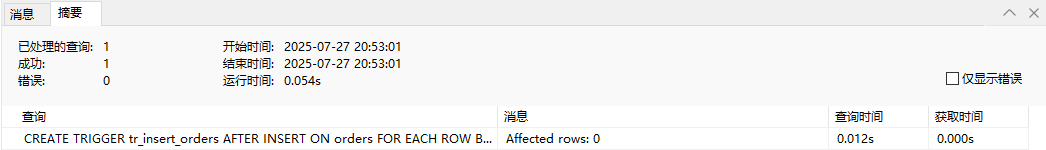
WHERE orders.order\_id = NEW.order\_id;

UPDATE members

SET points= points + 10

WHERE member\_id= New.member\_id;

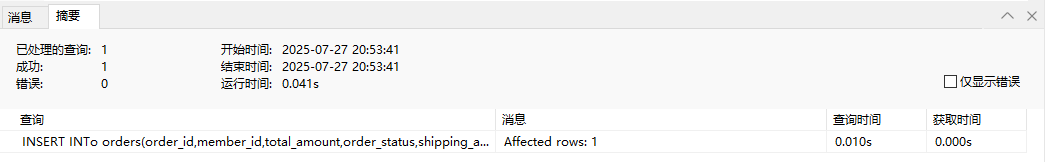
END;



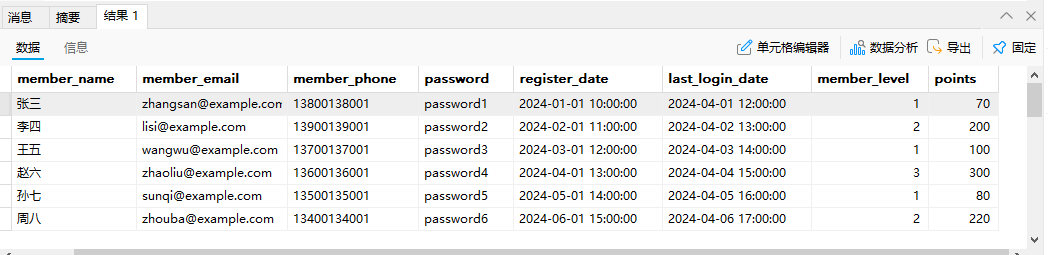
13.触发上面触发器

INSERT INTo orders(order\_id,member\_id,total\_amount,order\_status,shipping\_address,payment\_method)

VALUES(NULL,1,'100','未支付','四川','微信支付');



14.查看触发结果



15.事务

事务TRANSACTION：若干个增删改查 操作集合；

16.事务特性

ACID: 原子性 一致性 隔离性 持久性

17.事务的开启

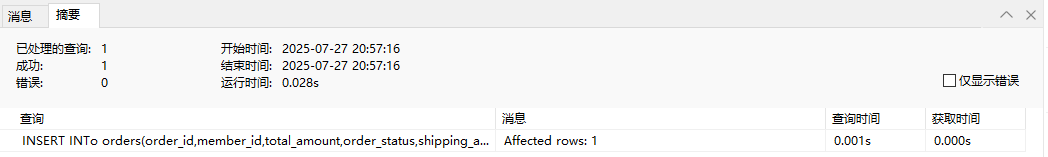
START TRANSACTION;

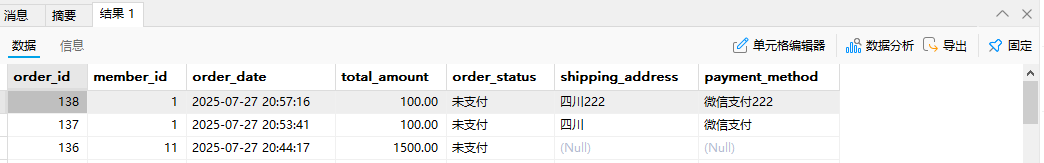


18.开启事务后进行操作并查询结果

INSERT INTo orders(order\_id,member\_id,total\_amount,order\_status,shipping\_address,payment\_method)

VALUES(NULL,1,'100','未支付','四川222','微信支付222');

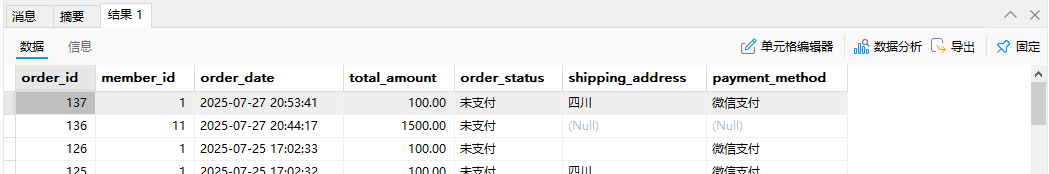




18.事务回滚后并查看

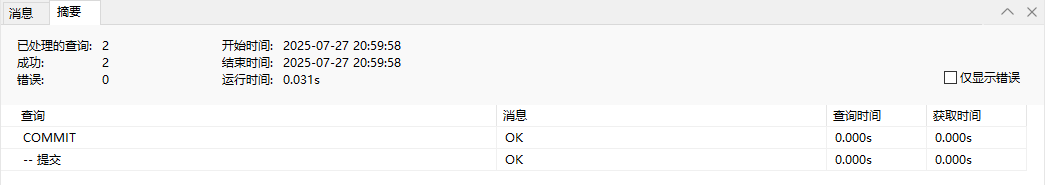
ROLLBACK; -- 回滚事务

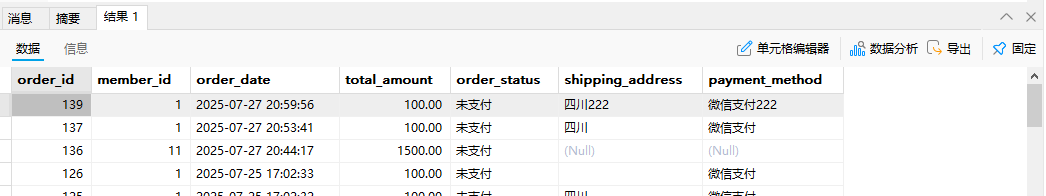




19.提交事务

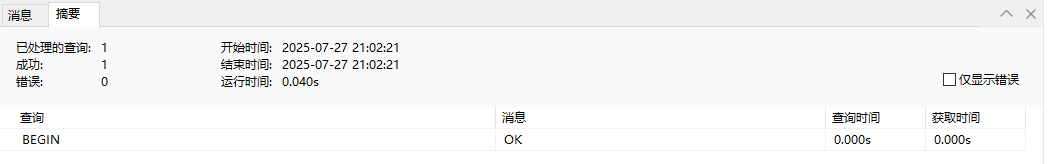
COMMIT; -- 提交





20.事务的另一种开启方式

BEGIN;



21.BEGIN方式插入数据并提交

INSERT INTO orders(order\_id,member\_id,total\_amount,order\_status,shipping\_address,payment\_method)

VALUES(NULL,'2',100,'未支付','四川成都','微信支付');

COMMIT ;





SELECT \* FROM orders ORDER BY order\_id DESC;

