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

实验报告

开课学期：2024 秋季

课程名称：数据库系统

实验名称：数据库设计

实验性质：设计型

实验学时：4 地点：T2210

学生班级：22 级 4 班

学生学号：220110430

学生姓名：吴梓滔

实验与创新实践教育中心制

2024 年 9 月

1 实验环境

请填写用到的操作系统和主要开发工具。

Windows11 操作系统、MySQL8.0 关系数据库管理系统、数据库设计工具 PowerDesigner16.5 汉化破解版。

2 实验过程

2.1 系统功能

*请结合文字、图表等方式清晰描述系统的功能。如有亮点功能请用*标志。*

校园食堂外送点单平台数据库系统能够支持校园食堂外送的业务场景，具有校园特色用户管理、菜品管理、订单管理、评论与评分、报表与分析等功能，并具有良好的安全性与数据完整性，满足校园场景中食堂运营、商家、师生用户同时在平台中的交易活动。

该系统根据业务场景对用户进行分类，为不同用户提供不同的操作权限，每类用户操作自己对应的业务对象，例如商家用户管理自己商铺，用户可以根据菜品下订单，但不能够修改菜品。*该数据库可以向不同用户提供扩展功能，例如通过存储过程和函数方便商家统计出月度收益，方便普通用户查询订单的信息，且基础表的设计尽可能精简轻量，通过设计不同的函数、过程来满足功能性需求。用户与业务场景的分类对应表如下。

用户类型	业务场景	操作的权限
普通用户	浏览菜品，创建外送订单，对订单发表评论	个人信息，订单信息，评论信息
商家	经营商铺，发布与维护菜品信息，更新商铺信息，接收并更新订单状态	个人信息，商铺，菜品，报表，订单信息
食堂管理员	管理食堂，更新食堂的相关信息	个人信息，食堂信息

数据库还实现了用户点评的功能，并且*对商家进行了保护，防止恶意差评。通过设置触发器，用户只能对自己创建且已经完成的订单发布评论，杜绝了未消费进行恶意评论的情况。

2.2 数据库设计

2.1.1 ER 图

要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。

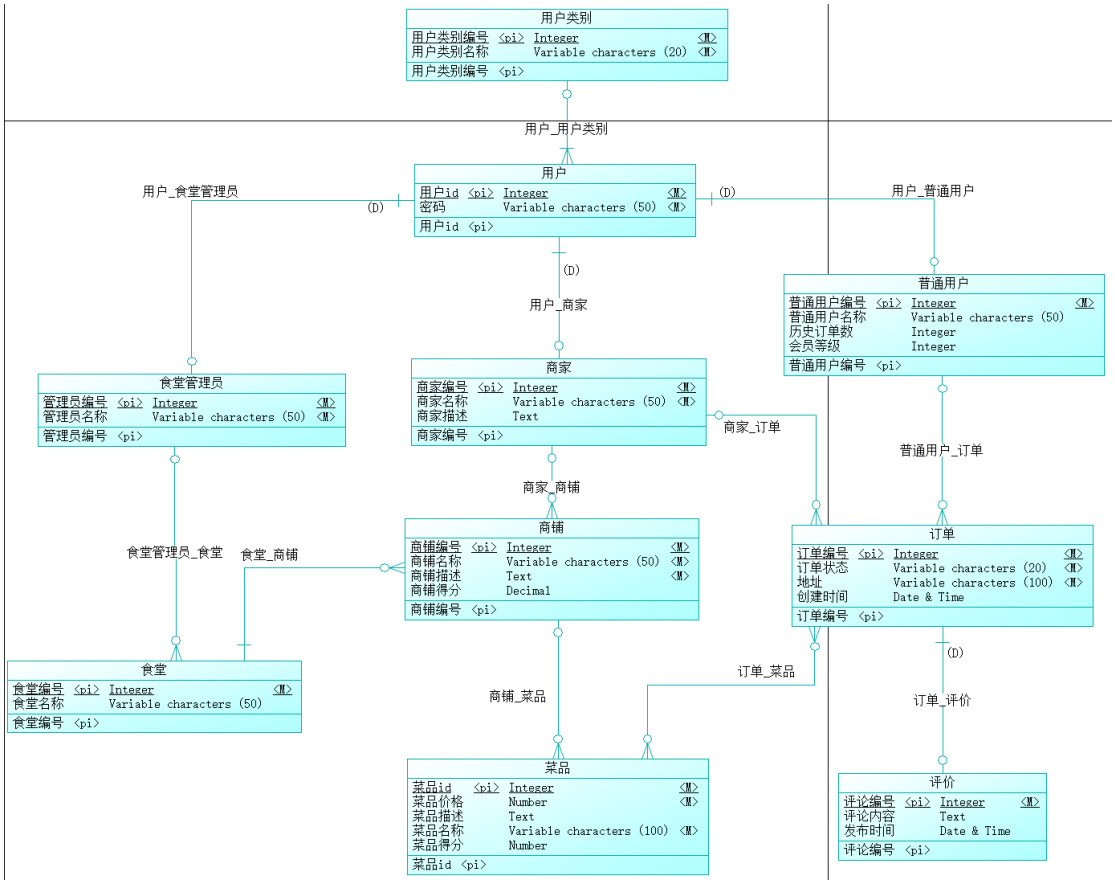


图 2.1 ER 图

2.1.2 LDM 图

要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。

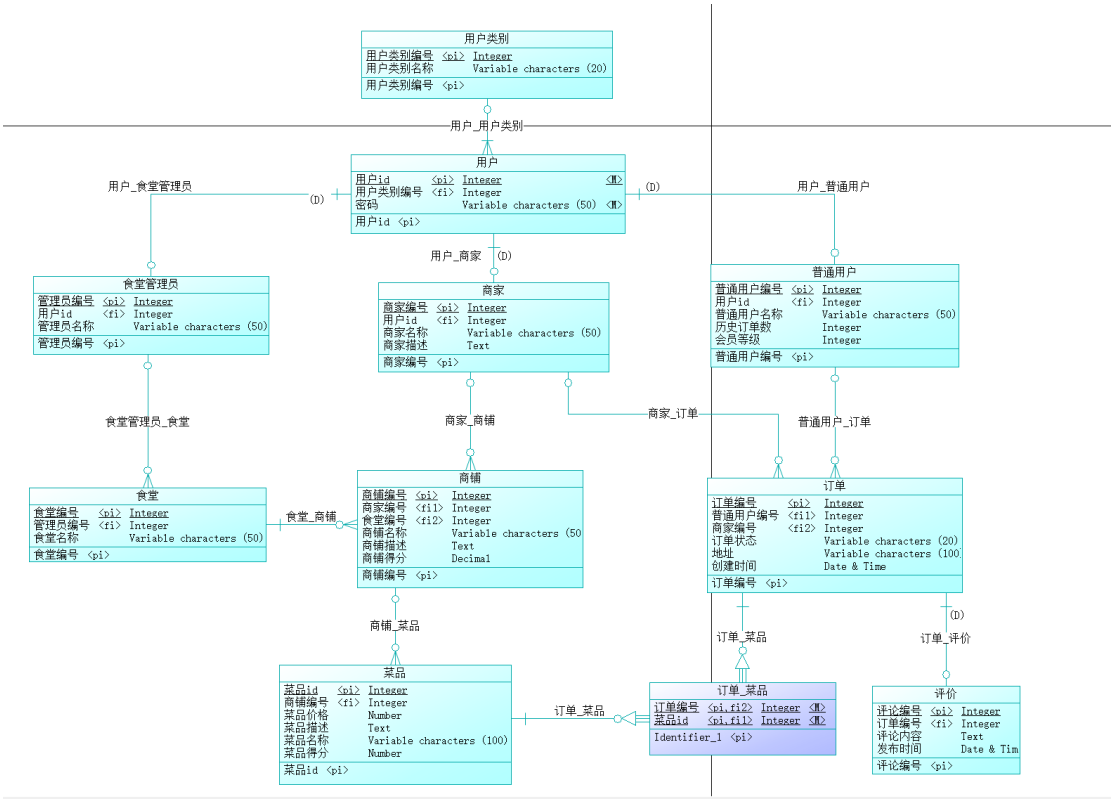


图 2.2 LDM 图

2.1.3 PDM 图

要求：截图务必清晰，如果图太大可截图一个总图，然后再分块截图。如果看不清截图会影响成绩。

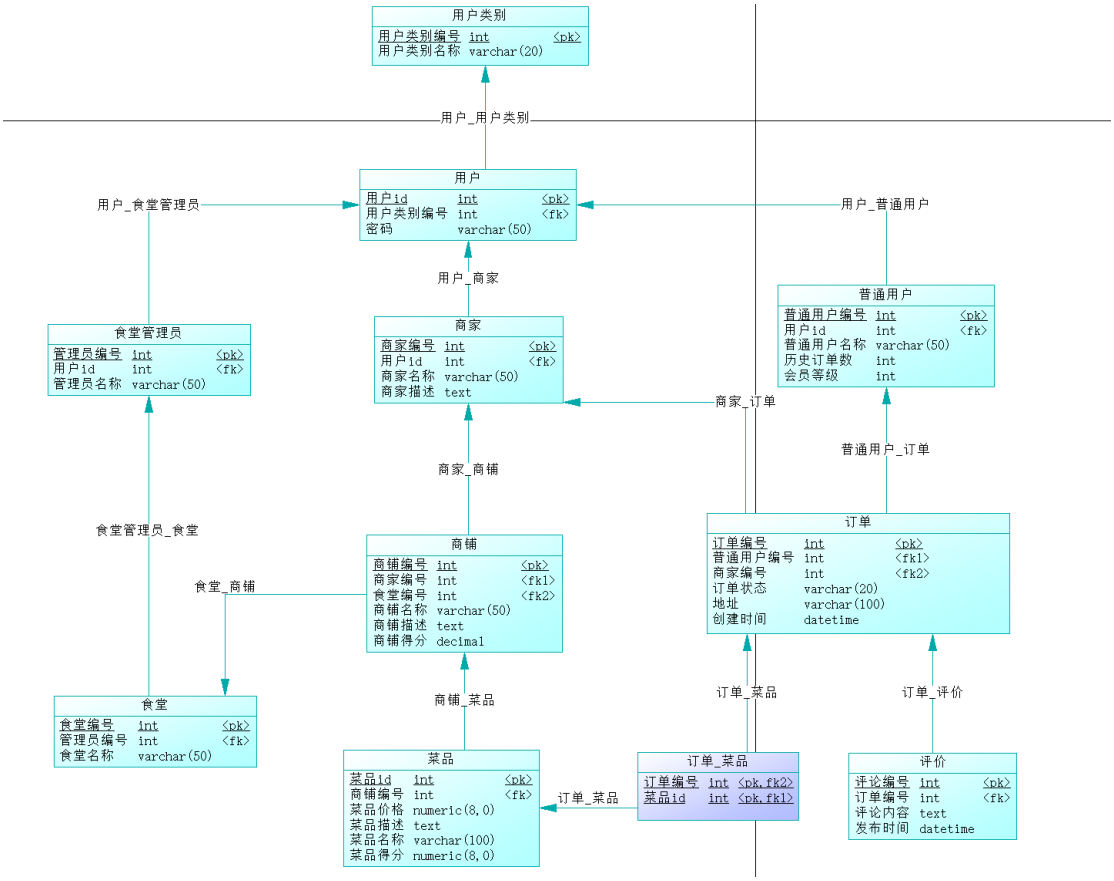


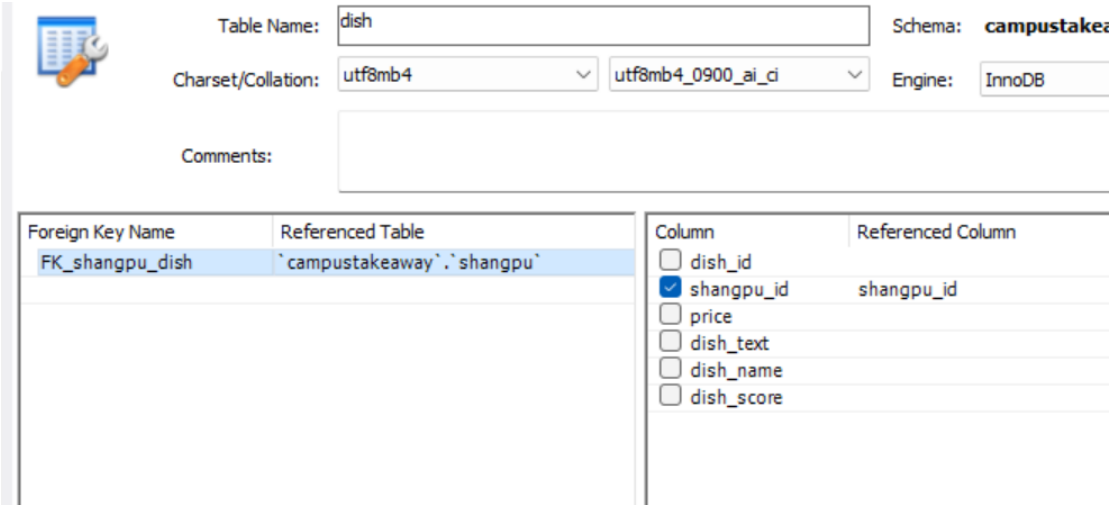
图 2.3 PDM 图

2.1.4 数据库表结构

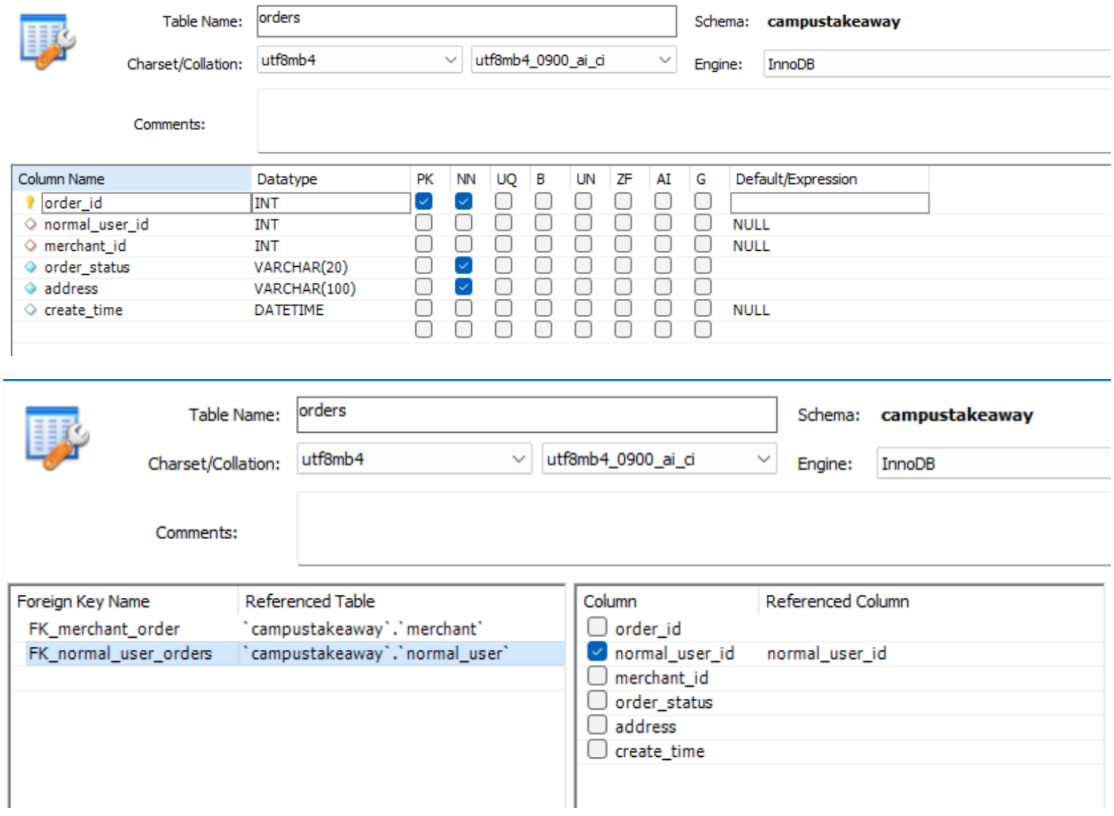
1、 表结构

选取 2-3 个比较有代表性的表结构截图，体现主键约束、外键约束、空值约束等。

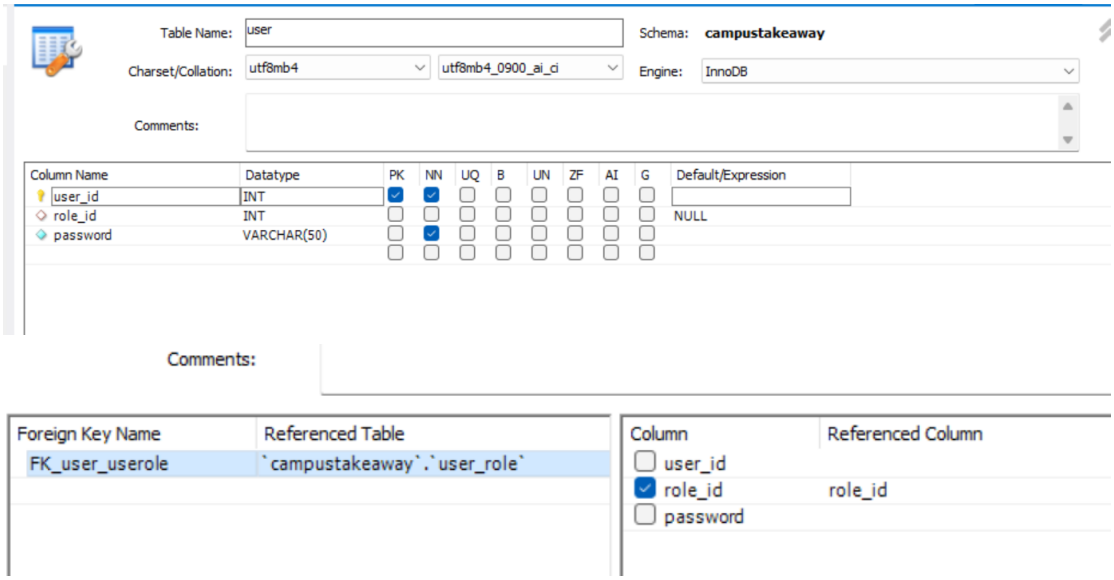
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
dish_id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
shangpu_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
price	DECIMAL(8,0)	☐	☑	☐	☐	☐	☐	☐	☐	
dish_text	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
dish_name	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
dish_score	DECIMAL(8,0)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	



dish 表，主键为 dish_id,能唯一标识某个商铺的一个菜品；外键 shangpu_id，是引用自商铺的表 shangpu。菜品的 dish_id,shangpu_id,price,dish_name 即编号，所属商铺编号，价格和名称不允许为空。



orders 表，主键为 order_id,唯一标识一个订单。外键有 merchant_id 和 normal_user_id 分别来自于 merchant 表和 normal_user 表。规定了订单的状态、送餐地址、id 不允许为空。

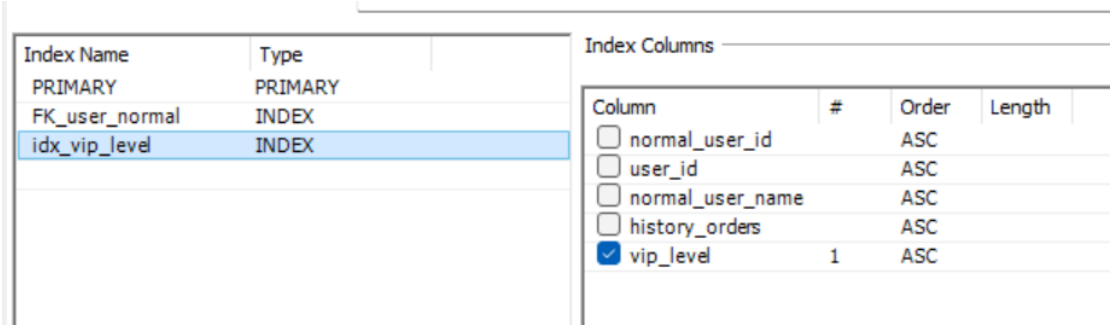


user 表，主键为 user_id，是用户注册时自己创建的帐号，不允许相同。外键有 role_id，引用自 user_role 表，记录每个用户的类型。空值约束有 password，密码不能为空。

2、 索引

1) 索引截图

通过 CREATE INDEX idx_vip_level ON normal_user (vip_level);语句，在普通用户表的会员等级 vip_level 列上创建了索引。



2) 使用场景说明（用途）

在用户量大，并且对普通用户根据下订单的数量进行会员等级分级后，经常需要查询不同等级会员用户来做一些专门的操作。对用户的会员等级做一个索引，会使查询某等级会员的速度更快。

3、 视图

1) 视图截图

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `view_order_comments` AS
    SELECT
        `o`.`order_id` AS `order_id`,
        `m`.`merchant_name` AS `merchant_name`,
        `nu`.`normal_user_name` AS `normal_user_name`,
        `c`.`comment_text` AS `comment_text`,
        `o`.`create_time` AS `order_create_time`,
        `c`.`comment_create_time` AS `comment_create_time`
    FROM
        (((`comment` `c`
        JOIN `orders` `o` ON ((`c`.`order_id` = `o`.`order_id`)))
        JOIN `merchant` `m` ON ((`o`.`merchant_id` = `m`.`merchant_id`)))
        JOIN `normal_user` `nu` ON ((`o`.`normal_user_id` = `nu`.`normal_user_id`)))
```

图 3.1 视图的创建代码

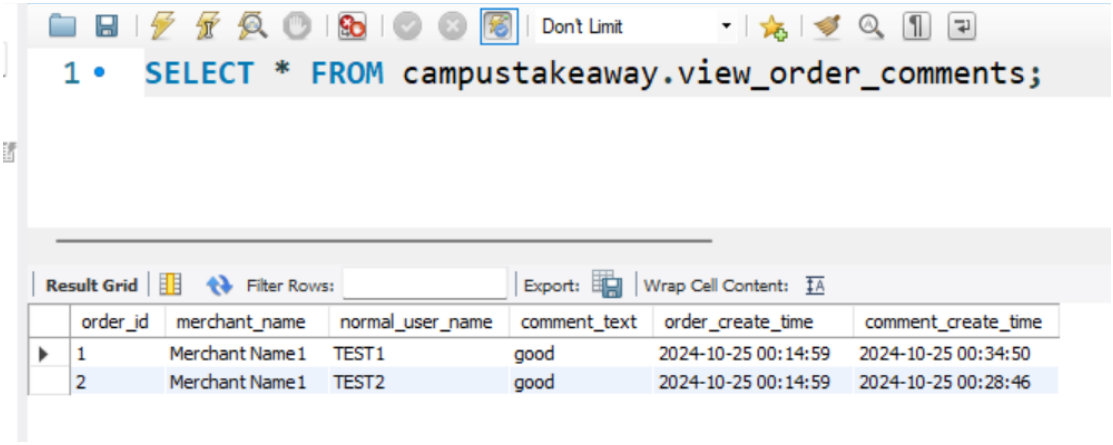


图 3.2 查询视图的结果

2) 使用场景说明（用途）

用于展示评论。将订单的商家名、顾客用户名、评论内容、创建订单的时间和发表评论的时间集中在一起，构成用于展示的数据。应用时可以将该视图转换成完整的评论条目。

4、 触发器

1) 触发器截图

图 4.1 comment 表上检查订单是否可以评论的触发器

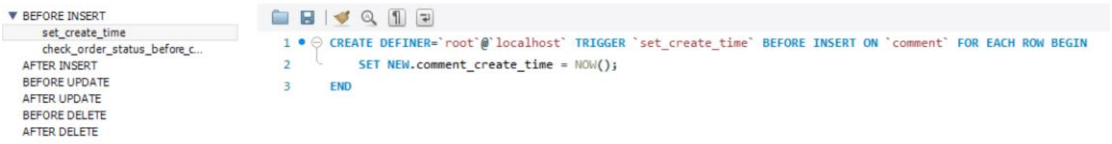


图 4.2 comment 表上自动设置创建时间的触发器

```
CREATE DEFINER=`root`@`localhost` TRIGGER `set_ordercreate_time` BEFORE INSERT ON `orders` FOR EACH ROW BEGIN
    SET NEW.create_time = NOW();
END
```

图 4.3 orders 表上自动设置创建时间的触发器

2)　　使用场景说明（用途）

当试图创建评论时，会在 comment 表插入之前，检查要评论的那个订单是否已经完成。只有已完成的订单能够成功创建评论。这个功能是为了防止没有消费恶意刷评论，误导其他消费者。

此外，还有两个触发器分别建立在 comment 表和 orders 表上，作用是在创建评论或订单时自动将当前时间填入创建时间一列。

3)　　验证触发器

```
-- 5 创建订单
INSERT INTO orders (normal_user_id, merchant_id, order_status, address)
VALUES ( 1, 1, '已下单', 'A02');
```

图 4.4

	order_id	normal_user_id	merchant_id	order_status	address	create_time
▶	1	1	1	已下单	A02	2024-10-25 00:14:59

图 4.5

① 验证 orders 上自动设置创建时间的触发器。如图 4.4 使用上面没有指定时间的语句创建一个订单。随后查询 orders 表，如图 4.5，发现创建时间是做实验时的系统时间，说明触发器正常工作，自动调用系统时间填入 create_time 一列。

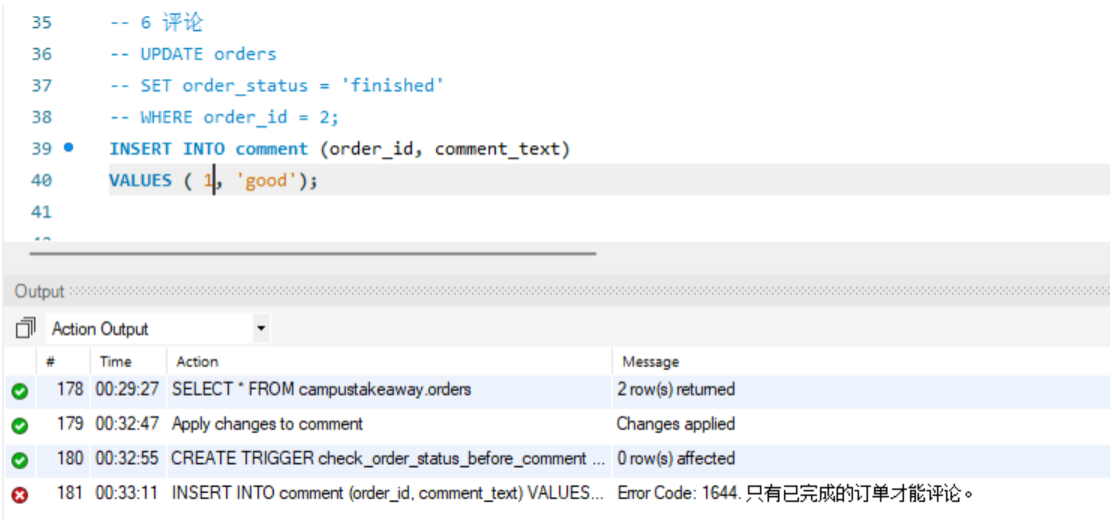


图 4.6

图 4.7

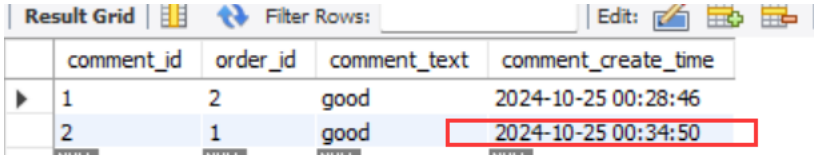


图 4.8

② 验证 comment 表上的两个触发器。如图 4.6，此时订单编号 order_id=1 的订单已经创建，但未完成，创建对订单 1 的评论，由于订单尚未完成，无法创建评论，输出了一个 error。如图 4.7 所示，更新 order_id=1 的订单状态为已完成，再执行插入评论，就成功了。如图 4.8 所示，查询评论表，order_id=1 的评论成功创建。并且可以看到评论时间和图 4.7 中语句执行的时间是对的上的，说明两个触发器都在正常工作。

5、 存储过程或存储函数

1) 存储过程或存储函数截图

```sql
CREATE DEFINER=`root`@`localhost` FUNCTION `order_price`(order_id INT) RETURNS decimal(10,2)
    DETERMINISTIC
BEGIN
    DECLARE total_price DECIMAL(10, 2);

    SELECT sum(d.price)
    INTO total_price
    FROM orders_dish od, dish d
    WHERE od.order_id = order_id and  od.dish_id = d.dish_id;

    RETURN IFNULL(total_price, 0);
END
```

图 5.1 存储函数 order_price

Name: get_merchant_orders_last_month

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_merchant_orders_last_month`(IN merchant_id INT)
BEGIN
    -- Declare variables for date range
    DECLARE start_date DATE;
    DECLARE end_date DATE;

    -- Set the date range to the last month
    SET start_date = DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
    SET end_date = CURDATE() + 1;   # 这里多数一天，否则当天不被包含

    -- Select the orders for the given merchant in the last month, along with their total price and creation time
    SELECT
        o.order_id,
        o.normal_user_id,
        order_price(o.order_id) AS total_price,
        o.create_time
    FROM  orders o
    WHERE o.merchant_id = merchant_id
          AND o.create_time BETWEEN start_date AND end_date;

END
```

图 5.2 存储过程 get_merchant_orders_last_month

```sql
CALL get_merchant_orders_last_month(1);
```

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

order_id	normal_user_id	total_price	create_time
1	1	20.00	2024-10-25 00:14:59
2	2	20.00	2024-10-25 00:14:59

图 5.3 存储过程的调用结果

2) 使用场景说明（用途）

图 5.2 所示存储过程用于列举指定的一个商家在过去一个月的所有订单，订单成交价格，顾客 id 以及订单创建时间。得到的查询结果可以用于商家统计过去一个月的收益。

在该存储过程中调用了图 5.1 的函数。这个函数的作用是计算某一个订单的价格。而在存储过程中，通过调用函数 order_price 来直接得到某一个订单的价格。
综上，该存储过程的使用场景是商家在统计收益时使用来得到一个月内的所有订单，方便他们进行后续的计算。