数据库原理与应用

《 美团外卖订单 》

项目报告

组名： 美团小组

成员列表：

学号： 190430107 姓名： 陆妍瑞

学号： 190430139 姓名： 樊英杰

学号： 190430132 姓名： 曹俊

学号： 190430137 姓名： 杨帆

完成时间 2019 年 6 月 5 日

 

|  |  |
| --- | --- |
| **数据库设计** | 设计背景：  人们的生活节奏越来越快，人们对饮食的质量和需求都越来越大，随着餐饮业正在飞速的发展，网上支付功能的日益强大，人们选择了更加方便的餐饮方式。由此外卖行业的应运而生，我们做的数据库就有有关外卖系统的。  众所周知，外卖服务，必须是用户，商家和配送员三方联系起来的，而在一定的地区限制的前提上，商家必须提供电话号码，地址，商家名等联系方式除此之外还要列出菜单以及菜品的价格等信息来供用户选择。配送员也需要有姓名，电话号码等信息，方便联系。而用户需要提供他们自身的联系方式，以及地址，方便配送员送单。  数据库设计概念:  模仿美团外卖的外卖流程，菜单等功能实现的数据库管理功能  设计数据库软件：  E-R图：draw.io  流程图：MindManager  数据库：SQLyog |

|  |  |
| --- | --- |
| **ER图、数据库表和关系模式** |  |

|  |  |
| --- | --- |
|  | **关系模式**  **地区表**（地区名称）  **用户表**（用户ID，地区名称，用户姓名，用户电话，用户地址，用户身份证号码）  **订单**（用户ID，商家ID，配送员ID，总价格，下单时间，订单状态）  **商家**（商家ID，地区名称，商家店名，商家电话，商家地址，商家的老板姓名，老板的身份证号码）  **菜品**（菜品ID，商家ID，菜品类别，菜品名称，价格）  **菜品类别汇总表**（菜品类别）  **配送员表**（配送员ID，地区名称，配送员姓名，配送员电话）  **用户等级表**（配送员ID，消费总额，vip等级）  **配送员等级表**（配送员ID，订单总数，等级）  注：横线为pk,波浪线为fk |
| **功能的分类与实现** | **信息完善**  1用户注册：用户通过添加必要的信息到数据库来完成注册。  2用户信息修改：用户根据自己的编号来修改除编号外的其他信息。  3用户可以修改自己的名字，并且查找名字的历史记录。  4用户通过id查找自己的信息。  5商家注册：商家通过添加必要的信息到数据库来完成注册。  6商家信息修改：商家根据自己的编号来修改除编号外的其他信息。  7商家通过id查找自己的信息。  8菜品录入：商家通过添加菜品必要的信息到数据库来完成菜品录入。  9菜品信息修改：商家根据菜品编号来修改菜品除编号外的其他信息。  10信息完善阶段：商家可以修改菜品的价格，并且可以查找价格的历史纪录  **订单实现**  1用户根据商家的地区来选取菜品/用户根据菜品分类来选择菜品。  2用户下单：用户通过添加订单信息到数据库来提交订单。  3计算每份订单总价钱，并且实现满减功能。  4根据配送员的地区来选择配送员。  5用户确认：用户通过修改订单状态来确认收货。  6取消订单：用户通过信息订单状态来退款（订单状态：已下单0，已收货1，已退款2）  **统计功能**  1计算每个月每个用户的订单数量，过滤条件：时间和订单状态。  2计算每个用户的消费总额，并且提高其会员等级。  3计算每个月每个配送员的送单数量，过滤条件：时间和订单状态。  4计算每个配送员总的送单数量，并且提高其员工级别。  5统计每个用户在每段时间内的所有订单总价。  **菜单功能**  1菜品通过分类的菜单栏。  2每个商家的菜单栏。    **权限设置**  用户权限  1.用户可以有用户表的所有权限  2.用户可以修改订单表中订单状态的权限  配送员权限  1.配送员可以有配送员表的所有权限  2.配送员可以修改订单表中订单状态的权限  商家权限  1.商家可以有商家表的所有权限  2.商家可以有菜品表的所有权限 |
| **优点和不足** | **优点**：  设计上：  1.考虑地区的约束  2.多种菜单，尽可能完整的模仿美团APP的功能  3.尽可能减少重复表的使用  4.增加订单状态更贴合实际  ……  数据表上：  1.所有的ID为自增  2.将电话号码，身份证号码设置唯一键  3.在订单表中增加时间列  4.身份证号码（实名制），电话号码，地址， 姓名不为空  ……  功能上：  1.功能多样  2.统计功能：用户可以查询所有的消费总额或 一段时间内的消费金额（配送员类似）  3.查询功能绝大多是用存储过程加游标根据ID 来查询信息。  ……  **错误与修改：**  设计数据库阶段   1. 设计订单表为实体表 2. 修改订单表：增加日期和订单状态 3. 删除重复表：顾客总表和骑手总表 4. 增加顾客vip表，骑手的等级表   实现数据库阶段：   1. 修改数据类型：价格为double 2. 增加游标在存储函数中遍历 3. 修改表的插入数据 4. 菜品价格的历史纪录：DROP TABLE IF EXISTS businformation; 5. 增加触发器：客户的姓名的历史纪录 6. 增加触发器：满10减1 7. 增加存储过程：用户的消费总额和骑手的送单数来判断VIP等级和员工等级 8. 增加视图：不同菜单。   **不足：**  未实现功能：  1. 配送员超时或在规定时间送达的记录  2. 订单表与用户表，骑手表，商店表三个表的id同步  3. 在只能在同一家商店购买商品  4. 订单明细表  5. 累加计算总价格 |

|  |  |
| --- | --- |
| **附件SQL语句** | 建表/插入数据  -- 创建数据库  CREATE DATABASE takeaway;  USE takeaway;  -- 建表  #地区表  CREATE TABLE `area`(  areaName VARCHAR(20) PRIMARY KEY  )ENGINE=INNODB DEFAULT CHARSET utf8;  #用户表  CREATE TABLE customer(  cusId INT PRIMARY KEY AUTO\_INCREMENT,  areaName VARCHAR(20),  cusName VARCHAR(10) NOT NULL,  cusPhone CHAR(12) NOT NULL,  cusAddress VARCHAR(256) NOT NULL,  cusIdCard CHAR(18) NOT NULL,  CONSTRAINT cus\_areaName\_fk FOREIGN KEY(areaName) REFERENCES `area`(areaName)  )ENGINE=INNODB DEFAULT CHARSET utf8;  #创建用户的会员等级表  CREATE TABLE vip\_customer(cusId INT ,  sum\_price DOUBLE,  vip\_name VARCHAR(6),  CONSTRAINT cus\_vip\_fk FOREIGN KEY(cusid) REFERENCES customer(cusid)  );  #创建配送员的送单数量表  CREATE TABLE orderNum\_ridder(  ridId INT ,  sum\_order DOUBLE,  good\_ridder CHAR(10),  CONSTRAINT num\_rider\_fk FOREIGN KEY(ridId) REFERENCES rider(ridId)  );  #商家表  CREATE TABLE business(  busiId INT PRIMARY KEY AUTO\_INCREMENT,  areaName VARCHAR(20),  busiName VARCHAR(10) NOT NULL,  busiPhone CHAR(12),  busiAddress VARCHAR(256) NOT NULL,  busiBossName VARCHAR(10),  busiBossIdCard CHAR(18) NOT NULL,  CONSTRAINT busi\_areaName\_fk FOREIGN KEY(areaName) REFERENCES `area`(areaName)  )ENGINE=INNODB DEFAULT CHARSET utf8;  #菜品类别汇总表  CREATE TABLE foodsCategory(  fooCategory VARCHAR(10) PRIMARY KEY  )ENGINE=INNODB DEFAULT CHARSET utf8;  #菜品表  CREATE TABLE foods(  fooId INT PRIMARY KEY AUTO\_INCREMENT,  busiId INT,  fooCategory VARCHAR(10),  fooName VARCHAR(15) NOT NULL,  fooPrice DOUBLE NOT NULL,  CONSTRAINT foo\_busiId\_fk FOREIGN KEY(busiId) REFERENCES business(busiId)  )ENGINE=INNODB DEFAULT CHARSET utf8;    #配送员表  CREATE TABLE rider(  ridId INT PRIMARY KEY AUTO\_INCREMENT,  areaName VARCHAR(20) ,  ridName VARCHAR(10) NOT NULL,  ridPhone CHAR(12) NOT NULL,  CONSTRAINT rid\_areaName\_fk FOREIGN KEY(areaName) REFERENCES `area`(areaName)  )ENGINE=INNODB DEFAULT CHARSET utf8;  #订单表  CREATE TABLE orders(  cusId INT,  busiId INT,  ridId INT,  totalPrice DOUBLE(10,2) DEFAULT 0,  PRIMARY KEY(cusId,busiId,ridId),  CONSTRAINT ord\_cusId\_fk FOREIGN KEY(cusId) REFERENCES customer(cusId),  CONSTRAINT ord\_busiId\_fk FOREIGN KEY(busiId) REFERENCES business(busiId),  CONSTRAINT ord\_ridId\_fk FOREIGN KEY(ridId) REFERENCES rider(ridId)  )ENGINE=INNODB DEFAULT CHARSET utf8;  #增加列:下单时间  ALTER TABLE orders ADD order\_time TIMESTAMP DEFAULT NOW() NOT NULL;  #增加列:订单状态  ALTER TABLE orders ADD state INT DEFAULT 0 NOT NULL COMMENT '已下单0,已收货1,已退款2';  -- 索引  CREATE UNIQUE INDEX cus\_cusIdCard\_inx ON customer(cusIdCard);  CREATE UNIQUE INDEX cus\_cusIdPhoen\_inx ON customer(cusPhone);  CREATE UNIQUE INDEX busi\_busiIdPhone\_inx ON business(busiPhone);  CREATE UNIQUE INDEX busi\_busiIdCard\_inx ON business(busiBossIdCard);  -- 插入数据  #向地区表插入数据  INSERT INTO AREA VALUES  ('南通'),('南京'),('徐州'),('无锡');  #向商家表中插入数据  INSERT INTO business VALUES  (1,'南通','阿宝海鲜排档','15175461131','南通 崇川区 江苏省环城南路1号西北方向158米','王铁根','320124190702035011'),  (2,'南通','阿福砂锅','15175461138','南通 崇川区 南大街八仙城西门包福记隔壁','仇铁牛','320124190002033018'),  (3,'南京','阿辉好宴世家','15175461132','鼓楼区湖南路狮子桥2号湖南路步行街内(近湖北路)','周大树','360124190002035012'),  (4,'南京','南京大排档','15175461136','鼓楼区湖南路狮子桥2号(湖北路口)','谢大海','320124190502035016'),  (5,'徐州','徐州饭店','15175461134','江苏省徐州市淮海东路201-43','程孝先','320124190002035014'),  (6,'无锡','好家快捷酒店','15175461133','无锡市太湖大道旁塘南路98号','曹好看','320124190002035093'),  (7,'无锡','莫泰连锁旅店','15175461135','无锡崇安区北大街1号A座','刘法','320124190002035055'),  (8,'无锡','如家快捷酒店','15175461137','无锡市五爱路81号','马连良','320124190002035057');  #向用户信息表中插入数据  INSERT INTO customer VALUES  (1,'南京','赵大华','13378446985','秦淮区 大行宫新世纪广场东门对面科巷新寓二栋103室','320646199901025012'),  (2,'南京','张石山','13378446989','秦淮区 大行宫科巷菜场西门','320646199901025015'),  (3,'南通','王子久','13378446981','南通市 如皋市 中山南路','320646199901025011'),  (4,'无锡','张成基','13378446983','无锡市环湖蠡园','320646199901025013'),  (5,'无锡','孙天民','13378446986','无锡市站北路98号','320646199901025016'),  (6,'徐州','张小龙','13378446987','徐州市新城区昆仑大道1号','320646199901025014'),  (7,'徐州','蔡德霖','13378446973','徐州新城区大龙湖附近','320646199901025017');  #向菜品表中插入数据  INSERT INTO foods VALUES  (1,1,'小吃','牛肉锅贴',18),  (2,1,'主食','鸭血粉丝',20),  (3,1,'甜点','桂花糕',12),  (4,1,'饮料','酸梅汤',5),  (5,2,'小吃','煮干丝',12),  (6,2,'主食','小笼包',10),  (7,2,'甜点','豆腐脑',6),  (8,2,'饮料','可乐',5),  (9,3,'小吃','状元豆',19),  (10,3,'主食','葱油饼',20),  (11,3,'甜点','草莓果仁酸奶',12),  (12,3,'饮料','桂花糖芋苗',15),  (13,4,'小吃','姜母鸭',38),  (14,4,'主食','花卷',10),  (15,4,'甜点','冰冻葡萄',12),  (16,4,'饮料','番茄芒果汁',9),  (17,5,'小吃','五香卷',8),  (18,5,'主食','肉三鲜馄饨',20),  (19,5,'甜点','巧克力蛋糕',19),  (20,5,'饮料','鲜榨粒粒橙',10);  #菜品类别表中插入数据  INSERT INTO `foodscategory` VALUES  ('小吃'),('主食'),('甜品'),('饮料'),('休闲食品');  #向配送员表中插入数据  INSERT INTO rider VALUES  (1,'南通','章汉夫','1512477461'),  (2,'南通','宗敬先','13812577476'),  (3,'南通','高尚德','13812457466'),  (4,'南京','赵进喜','13885477462'),  (5,'南京','谭平山','13212477467'),  (6,'南京','汤绍箕','13812477463'),  (7,'徐州','胡宝善','13852477465'),  (8,'徐州','王子久','13852477467'),  (9,'徐州','孙红雷','13817487467'),  (10,'无锡','黄磊','13812459569'),  (11,'无锡','黄渤','13812457411'),  (12,'无锡','王迅','13816477411');  #向订单表中插入数据  INSERT INTO orders(cusId,busiId,ridId) VALUES  (1,1,1),  (2,1,9),  (2,2,2),  (2,3,1),  (2,5,1),  (3,2,10),  (3,3,3),  (4,3,1),  (4,4,4),  (5,3,1),  (5,4,2),  (5,5,5),  (6,5,3);  权限设置  /\*  用户权限  1.用户可以有用户表的所有权限  2.用户可以修改订单表中订单状态的权限  \*/  #创建用户  CREATE USER 'mt4'@'localhost' IDENTIFIED BY'1234';  ALTER USER 'mt4'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '1234';  #创建角色并赋予权限  CREATE role cus;  GRANT ALL PRIVILEGES ON `customer`.\* TO cus;  GRANT UPDATE(state) ON takeaway.`orders` TO cus;  #将角色赋给用户  GRANT cus TO 'mt4'@'localhost' ;  /\*  配送员权限  1.配送员可以有配送员表的所有权限  2.配送员可以修改订单表中订单状态的权限  \*/  #创建配送员  CREATE USER 'rider1'@'localhost' IDENTIFIED BY'1234';  ALTER USER 'rider1'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '1234';  #创建角色并赋予权限  CREATE role rid;  GRANT ALL PRIVILEGES ON `rider`.\* TO rid;  GRANT UPDATE(state) ON takeaway.`orders` TO rid;  #将角色赋给配送员  GRANT rid TO 'rider1'@'localhost';  /\*  商家权限  1.商家可以有商家表的所有权限  2.商家可以有菜品表的所有权限  \*/  #创建商家  CREATE USER 'bus1'@'localhost' IDENTIFIED BY'1234';  ALTER USER 'bus1'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '1234';  #创建角色并赋予权限  CREATE role bus;  GRANT ALL PRIVILEGES ON foods.\* TO bus;  GRANT ALL PRIVILEGES ON business.\* TO bus;  #将角色赋给商家  GRANT bus TO 'bus1'@'localhost';  注册/修改信息  -- 注册，修改，查看信息  #注册用户信息  DELIMITER $$  CREATE PROCEDURE addCustomer(IN cusid INT(11),IN areaname VARCHAR(20),cusname VARCHAR(20),cusphone CHAR(12),cusaddress VARCHAR(256),cusidcard CHAR(18))  BEGIN  INSERT INTO customer VALUES(cusid,areaname,cusname,cusphone,cusaddress,cusidcard);  END $$  DELIMITER ;  CALL addCustomer(19,'南通','成辰就','13545788941','南通市如皋市白蒲镇','320589825402035401');  #客户信息修改  DELIMITER $$  CREATE PROCEDURE updateCustomer(IN up\_cusid INT,IN up\_areaname VARCHAR(10),IN up\_cusname VARCHAR(10),IN up\_cusphone VARCHAR(12),IN up\_cusaddress VARCHAR(256),IN up\_cusidcard CHAR(18))  BEGIN  UPDATE customer SET areaname=up\_areaname,cusname=up\_cusname,cusPhone=up\_cusphone,cusAddress=up\_cusAddress,cusIdCard=up\_cusIdCard WHERE cusid=up\_cusid;  END $$  DELIMITER ;  CALL updateCustomer(17,'南京','王虎','15678414568','南京市江宁区江宁大学城','330252198506035263');  #查看用户信息  DELIMITER $$  CREATE PROCEDURE cus\_information(IN up\_cusid INT)  BEGIN  DECLARE cus\_name CHAR(10);  DECLARE cus\_phone CHAR(20);  DECLARE cus\_address VARCHAR(256);  DECLARE cus\_Idcard CHAR(20);  SET cus\_name=(SELECT cusName FROM customer WHERE cusID=up\_cusid);  SET cus\_phone=(SELECT cusphone FROM customer WHERE cusID=up\_cusid);  SET cus\_address=(SELECT cusaddress FROM customer WHERE cusID=up\_cusid);  SET cus\_Idcard=(SELECT cusIdcard FROM customer WHERE cusID=up\_cusid);  DROP TABLE IF EXISTS cusinformation;  CREATE TABLE cusinformation(NAME CHAR(10),phone CHAR(20),address VARCHAR(256),idcard CHAR(20));  INSERT INTO cusinformation VALUES(cus\_name,cus\_phone,cus\_address,cus\_idcard);  SELECT \* FROM cusinformation;    END $$  DELIMITER ;  CALL cus\_information(2);  #注册商家信息  DELIMITER $$  CREATE PROCEDURE addBusiness(IN busiid INT(11),IN areaname VARCHAR(20),businame VARCHAR(20),busiphone CHAR(12),busiaddress VARCHAR(256),busibossname VARCHAR(10),busibossidcard CHAR(18))  BEGIN  INSERT INTO business VALUES(busiid,areaname,businame,busiphone,busiaddress,busibossname,busibossidcard);  END $$  DELIMITER ;  CALL addBusiness(9,'南京','云月楼','15546462122','南京市栖霞区仙林大学城','朱时茂','521653108501035201');  #商家信息修改  DELIMITER $$  CREATE PROCEDURE updateBusiness(IN up\_busiid INT,IN up\_areaname VARCHAR(10),IN up\_businame VARCHAR(10),IN up\_busiphone VARCHAR(12),IN up\_busiaddress VARCHAR(10),IN up\_busibossname VARCHAR(10),IN up\_bossidcard CHAR(18))  BEGIN  UPDATE business SET areaname=up\_areaname,businame=up\_businame,busiPhone=up\_busiphone,busiAddress=up\_busiAddress,busibossname=up\_busibossname,busibossidcard=up\_bossidcard WHERE busiid=up\_busiid;  END $$  DELIMITER ;  CALL updateBusiness(1,'南通','酒香楼','15678414564','南通市长江路32号','郭京飞','310512197704056541');  #查看商家信息  DELIMITER $$  CREATE PROCEDURE busi\_information(IN up\_busiId INT)  BEGIN  DECLARE busi\_name CHAR(10);  DECLARE busi\_phone CHAR(20);  DECLARE busi\_address VARCHAR(256);  DECLARE busi\_bossname VARCHAR(10);  DECLARE busi\_Idcard CHAR(20);  SET busi\_name=(SELECT busiName FROM business WHERE busiID=up\_busiid);  SET busi\_phone=(SELECT busiphone FROM business WHERE busiID=up\_busiid);  SET busi\_address=(SELECT busiaddress FROM business WHERE busiID=up\_busiid);  SET busi\_bossname=(SELECT busibossname FROM business WHERE busiID=up\_busiid);  SET busi\_Idcard=(SELECT busiBossIdcard FROM business WHERE busiID=up\_busiid);  DROP TABLE IF EXISTS businformation;  CREATE TABLE businformation(NAME CHAR(10),phone CHAR(20),address VARCHAR(256),bossname CHAR(10),idcard CHAR(20));  INSERT INTO businformation VALUES(busi\_name,busi\_phone,busi\_address,busi\_bossname,busi\_idcard);  SELECT \* FROM businformation;    END $$  DELIMITER ;  CALL busi\_information(4);  #存储过程:菜品录入  DELIMITER $$  CREATE PROCEDURE addFoods(IN fooid INT,IN busiid INT,IN fooCategory VARCHAR(10),fooname VARCHAR(10),fooprice DOUBLE)  BEGIN  INSERT INTO foods VALUES(fooid,busiid,foocategory,fooname,fooprice);  END $$  DELIMITER ;  CALL addFoods(21,4,'休闲食品','香瓜子',15);  #存储过程:菜品信息修改  DELIMITER $$  CREATE PROCEDURE updateFoods(IN up\_fooid INT,IN up\_busiid INT,IN up\_fooCategory VARCHAR(10),IN up\_fooname VARCHAR(10),IN up\_fooprice DOUBLE)  BEGIN  UPDATE foods SET busiid=up\_busiid,fooCategory=up\_fooCategory,fooName=up\_fooname,fooPrice=up\_fooPrice WHERE fooid=up\_fooid;  END $$  DELIMITER ;  CALL updateFoods(3,1,'甜点','朱古力豆',21);  -- 历史记录  -- 触发器  #订单计算总价  DELIMITER $$  USE `takeaway`$$  CREATE TRIGGER `tri\_supportProduct` BEFORE INSERT ON `orders` FOR EACH ROW  BEGIN  DECLARE tPrice DECIMAL(10,2) DEFAULT 0;  SELECT SUM(fooprice) INTO tPrice  FROM business NATURAL JOIN foods  WHERE business.busiid=new.busiid;  SET new.totalPrice=tPrice;  IF tPrice>=10 THEN  SET new.totalprice=tPrice-1;  END IF;  END $$  DELIMITER ;  #各个商家的的地区  CREATE VIEW business\_area  AS  SELECT busiName,areaname  FROM business;  #各配送员的地区  CREATE VIEW rider\_area  AS  SELECT ridName,areaName FROM rider;  #修改用户姓名的历史记录  DROP TABLE IF EXISTS oldCusName;  CREATE TABLE oldCusName  AS  SELECT cusid,cusname  FROM `customer`;  TRUNCATE TABLE oldCusName;  DROP TABLE IF EXISTS newCusName;  CREATE TABLE newCusName  AS  SELECT cusid,cusname  FROM `customer`;  TRUNCATE TABLE newCusName;  DELIMITER $  USE `takeaway`$  CREATE TRIGGER cus\_modify\_name  AFTER UPDATE ON `customer`  FOR EACH ROW  BEGIN  INSERT INTO oldCusName(cusid,cusname)  VALUES(old.cusid,old.cusname);  INSERT INTO newCusName(cusid,cusname)  VALUES(new.cusid,new.cusname);  END$  DELIMITER ;  UPDATE `customer`  SET cusname='孙尚香'  WHERE cusId=7;  SELECT \* FROM oldCusName;  SELECT\* FROM newCusName;  订单功能  -- 外卖流程:  #用户根据地区来选择商家  DELIMITER $  USE `takeaway`$  CREATE PROCEDURE area\_select\_busid(IN `area` VARCHAR(20))  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE busi\_id INT;  DECLARE bus\_all CURSOR FOR  SELECT busiid  FROM business  WHERE areaName=`area`;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  DROP TABLE IF EXISTS testcur;  CREATE TABLE testcur(busId INT);  OPEN bus\_all;  loopbusiness:LOOP  FETCH bus\_all INTO busi\_id;  IF done THEN  LEAVE loopbusiness;  END IF;    INSERT INTO testcur VALUES(busi\_id);  END LOOP;  CLOSE bus\_all;  SELECT \* FROM testcur;  END$  DELIMITER ;  CALL area\_select\_busid('南通');  #用户根据商家id来查询这个商家的菜单  DELIMITER $  USE `takeaway`$  CREATE PROCEDURE busiid\_select\_foods(IN bus\_id INT)  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE food\_id INT;  DECLARE food\_category VARCHAR(11);  DECLARE food\_name VARCHAR(20);  DECLARE food\_price DOUBLE;  DECLARE food\_all CURSOR FOR  SELECT fooId,fooCategory,fooName,fooprice  FROM foods  WHERE busiId=bus\_id;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  DROP TABLE IF EXISTS testcur;  CREATE TABLE testcur(foodId INT,foodCategory VARCHAR(11),foodName VARCHAR(20),foodPrice DOUBLE);  OPEN food\_all;  loopfood:LOOP  FETCH food\_all INTO food\_id,food\_category,food\_name,food\_price;  IF done THEN  LEAVE loopfood;  END IF;    INSERT INTO testcur VALUES(food\_id,food\_category,food\_name,food\_price);  END LOOP;  CLOSE food\_all;  SELECT \* FROM testcur;  END$  DELIMITER ;  CALL busiid\_select\_foods(1);  #或者用户根据菜的类别来选择菜名  DELIMITER $  USE `takeaway`$  CREATE PROCEDURE fc\_select\_foods(IN foo\_cate VARCHAR(20))  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE food\_id INT;  DECLARE food\_name VARCHAR(20);  DECLARE food\_price DOUBLE;  DECLARE food\_all CURSOR FOR  SELECT fooId,fooName,fooprice  FROM foods  WHERE fooCategory=foo\_cate;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  DROP TABLE IF EXISTS testcur;  CREATE TABLE testcur(foodId INT,foodName VARCHAR(20),foodPrice DOUBLE);  OPEN food\_all;  loopfood:LOOP  FETCH food\_all INTO food\_id,food\_name,food\_price;  IF done THEN  LEAVE loopfood;  END IF;    INSERT INTO testcur VALUES(food\_id,food\_name,food\_price);  END LOOP;  CLOSE food\_all;  SELECT \* FROM testcur;  END$  DELIMITER ;  CALL fc\_select\_foods('小吃');  #订单计算总价  DELIMITER $$  USE `takeaway`$$  CREATE TRIGGER `tri\_supportProduct` BEFORE INSERT ON `orders` FOR EACH ROW  BEGIN  DECLARE tPrice DECIMAL(10,2) DEFAULT 0;  SELECT SUM(fooprice) INTO tPrice  FROM business NATURAL JOIN foods  WHERE business.busiid=new.busiid;  SET new.totalPrice=tPrice;  IF tPrice>=10 THEN  SET new.totalprice=tPrice-1;  END IF;  END $$  DELIMITER ;  #客户下单：客户通过添加订单信息到数据库来提交订单  DELIMITER $$  CREATE PROCEDURE cus\_order(IN p\_cusid INT,IN p\_busiid INT,IN p\_ridid INT,IN p\_state INT)  BEGIN  INSERT INTO orders(cusid,busiid,ridid,state)VALUES (p\_cusid,p\_busiid,p\_ridid,p\_state);  END $$  DELIMITER ;  CALL cus\_order(1,2,5,0);  #根据地区查询配送员号（派遣订单）  DELIMITER $$  USE `takeaway` $$  CREATE PROCEDURE area\_select\_riderid(IN riderArea VARCHAR(20))  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE rider\_id VARCHAR(20);  DECLARE rider\_all CURSOR FOR  SELECT ridid  FROM rider  WHERE areaname=riderArea;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  DROP TABLE IF EXISTS testcur;  CREATE TABLE testcur(riderId INT);  OPEN rider\_all;  looprider:LOOP  FETCH rider\_all INTO rider\_id;  IF done THEN  LEAVE looprider;  END IF;    INSERT INTO testcur VALUES(rider\_id);  END LOOP;  CLOSE rider\_all;  SELECT \* FROM testcur;  END $$  DELIMITER ;  CALL area\_select\_riderid('南通');  #存储函数:查询用户在某店下的订单由谁配送  SET GLOBAL log\_bin\_trust\_function\_creators=TRUE;  DELIMITER $$  CREATE FUNCTION find\_ridid\_by\_cusid\_busiid(pcusid INT,pbusiid INT) RETURNS INT  BEGIN  RETURN(SELECT ridid FROM orders WHERE cusid=pcusid AND busiid=pbusiid);  END $$  DELIMITER ;  SELECT find\_ridid\_by\_cusid\_busiid(3,7);  #存储函数:根据订单id查询订单状态(0下单,1收货,2退款)  DELIMITER $$  USE takeaway $$  CREATE FUNCTION find\_state\_by\_id (cid INT,bid INT,rid INT) RETURNS INT  BEGIN  DECLARE v\_state INT;  SELECT state INTO v\_state FROM orders WHERE cusid=cid AND busiid=bid AND ridid=rid;  RETURN v\_state;  END $$  DELIMITER ;  SELECT find\_state\_by\_id(1,1,1);  #客户确认收货：客户通过修改订单状态来确认收货  DELIMITER $$  CREATE PROCEDURE confirmReceipt(IN p\_cusid INT,IN p\_busiid INT,IN p\_ridid INT)  BEGIN  UPDATE orders SET state=1 WHERE cusid=p\_cusid AND busiid=p\_busiid AND ridid=p\_ridid;  END $$  DELIMITER ;  CALL confirmReceipt(1,2,5);  #取消订单：客户通过修改订单状态来退款  DELIMITER $$  CREATE PROCEDURE cancellationOfOrder(IN p\_cusid INT,IN p\_busiid INT,IN p\_ridid INT)  BEGIN  UPDATE orders SET state=2 WHERE cusid=p\_cusid AND busiid=p\_busiid AND ridid=p\_ridid;  END $$  DELIMITER ;  CALL cancellationOfOrder(1,2,5);  #存储函数:根据配送员编号查询配送员姓名  DELIMITER $$  USE takeaway $$  CREATE FUNCTION find\_ridname\_by\_ridid(riderid INT) RETURNS CHAR(10)  BEGIN  DECLARE v\_ridname CHAR(10);  SELECT ridname INTO v\_ridname FROM rider WHERE ridid=riderid;  RETURN v\_ridname;  END $$  DELIMITER ;  SELECT find\_ridname\_by\_ridid(1);  #存储函数:根据商家id查询商家名称  DELIMITER $$  USE takeaway $$  CREATE FUNCTION find\_businame\_by\_busiid(busid INT) RETURNS VARCHAR(20)  BEGIN  DECLARE v\_name CHAR(20);  SELECT businame INTO v\_name FROM business WHERE busiid=busid;  RETURN v\_name;  END $$  DELIMITER ;  SELECT find\_businame\_by\_busiid(1);  #vip\_customer表累加总消费  DELIMITER $$  USE takeaway $$  CREATE TRIGGER calculate\_sumprice AFTER INSERT ON orders FOR EACH ROW  BEGIN  UPDATE vip\_customer SET sum\_price=sum\_price+new.totalPrice WHERE vip\_customer.`cusId`=new.cusid;  END $$  DELIMITER ;  SELECT \* FROM vip\_customer;  INSERT INTO orders(cusid,busiid,ridid) VALUES(2,5,7);  DELETE FROM orders WHERE cusid=1 AND busiid=1 AND ridid=10;  SELECT \* FROM orders;  SELECT \* FROM vip\_customer;  #ordernum\_ridder表累加总订单  DELIMITER $$  USE takeaway $$  CREATE TRIGGER calculate\_sumorder AFTER INSERT ON orders FOR EACH ROW  BEGIN  DECLARE tOrder INT DEFAULT 0;  SELECT COUNT(\*) INTO tOrder FROM orders GROUP BY ridid;  UPDATE ordernum\_ridder SET sum\_order=sum\_order+tOrder WHERE ordernum\_ridder.`ridId`=new.ridid;  END $$  DELIMITER ;  SELECT \* FROM orders;  SELECT \* FROM ordernum\_ridder;  INSERT INTO orders(cusid,busiid,ridid) VALUES(3,4,10);  DELETE FROM orders WHERE cusid=3 AND busiid=4 AND ridid=10;  其他功能  -- 菜单的实现功能  # 各个商家售卖的菜品名称  CREATE VIEW menuBar  AS  SELECT busiName,fooName,fooprice  FROM business JOIN foods;  # 按分类来查找菜品  CREATE VIEW cate\_foods  AS  SELECT busiName,fooName,fooprice  FROM business NATURAL JOIN foods  WHERE fooCategory='小吃';  -- 统计的功能  # 计算每个月每个客户的订单数量，过滤条件：时间和订单状态  CREATE VIEW cus\_buy  AS  SELECT cusId '客户id',COUNT(\*) '订单数量'  FROM orders  WHERE order\_time BETWEEN '2020-05-01' AND '2020-05-31' AND state IN(0,1)  GROUP BY cusid;  # 统计每个用户在每段时间内的所有订单总价  CREATE VIEW cus\_buyPrice  AS  SELECT cusId '客户id',SUM(totalPrice) '订单总价'  FROM orders  WHERE order\_time BETWEEN '2020-05-01' AND '2020-05-31' AND state=0  GROUP BY cusid;  # #通过订单向用户会员等级表中插入数据  INSERT INTO vip\_customer(cusId,sum\_price)  SELECT cusid,SUM(totalPrice)  FROM orders  WHERE state IN(0,1)  GROUP BY cusid;  #通过总消费来计算用户的vip等级  DELIMITER $  CREATE PROCEDURE vips\_customer(IN vipcusid INT)  BEGIN  DECLARE cus\_price DOUBLE;  DECLARE vip\_grade CHAR(20);  SET cus\_price=(SELECT sum\_price FROM `vip\_customer` WHERE cusid=vipcusid);  CASE  WHEN cus\_price<500 THEN SET vip\_grade='白银会员';  WHEN cus\_price<1000 AND cus\_price>=500 THEN SET vip\_grade='黄金会员';  ELSE SET vip\_grade='钻石会员';  END CASE;    UPDATE vip\_customer SET vip\_name=vip\_grade WHERE cusid=vipcusid;  END $  DELIMITER ;  CALL vips\_customer(6);  #计算每个配送员每个月的订单数，过滤条件：时间和订单状态  CREATE VIEW rider\_delivery  AS  SELECT ridid '配送员',COUNT(\*) '月订单数'  FROM orders WHERE order\_time BETWEEN '2020-05-01' AND '2020-05-31' AND state IN(0,1)  GROUP BY ridid;  DROP VIEW rider\_delivery;  # #通过订单想送货员数量表中插入数据  INSERT INTO orderNum\_ridder(ridId,sum\_order)  SELECT ridId ,COUNT(\*)  FROM orders  GROUP BY ridid;  #通过送单次数来评价配送员的等级  DELIMITER $  CREATE PROCEDURE ordersNum\_ridder(IN order\_ridid INT)  BEGIN  DECLARE rid\_order DOUBLE;  DECLARE rider\_grade CHAR(20);  SET rid\_order=(SELECT sum\_order FROM `ordernum\_ridder` WHERE ridid=order\_ridid);  CASE  WHEN rid\_order<5 THEN SET rider\_grade='白银员工';  WHEN rid\_order<10 AND rid\_order>=5 THEN SET rider\_grade='黄金员工';  ELSE SET rider\_grade='钻石员工';  END CASE;    UPDATE `ordernum\_ridder` SET good\_ridder=rider\_grade WHERE ridId=order\_ridid;  END $  DELIMITER ;  CALL ordersNum\_ridder(8);  -- 地区配送员接单排行榜  CREATE VIEW rider\_worknum  AS  SELECT areaname,ridid,ridname,COUNT(\*) AS 配送员送单数 FROM AREA NATURAL JOIN rider NATURAL JOIN orders GROUP BY ridid ORDER BY 配送员送单数 DESC;  事件  -- 事件  -- 每隔一天向vip\_customer表插入总订单数  DELIMITER $$  USE takeaway $$  CREATE PROCEDURE insert\_cus\_tOrders()  BEGIN  INSERT INTO vip\_customer(cusid,total\_orders)  SELECT cusid,COUNT(\*) FROM orders GROUP BY cusid;  END $$  DELIMITER ;  CREATE EVENT cus\_tOrders  ON SCHEDULE EVERY 1 DAY  STARTS NOW()  ENDS MAKEDATE(YEAR(NOW())+1,1)  ON COMPLETION PRESERVE  DO CALL insert\_cus\_tOrders;  SELECT \* FROM vip\_customer;  ALTER EVENT cus\_tOrders DISABLE;  ALTER EVENT cus\_tOrders ENABLE;  -- 每隔一天向ordernum\_ridder表插入总订单数  DELIMITER $$  USE takeaway $$  CREATE PROCEDURE insert\_rid\_tOrders()  BEGIN  INSERT INTO ordernum\_ridder(ridid,sum\_order)  SELECT ridid,COUNT(\*) FROM orders GROUP BY ridid;  END $$  DELIMITER ;  CREATE EVENT rid\_tOrders  ON SCHEDULE EVERY 1 DAY  STARTS NOW()  ENDS MAKEDATE(YEAR(NOW())+1,1)  ON COMPLETION PRESERVE  DO CALL insert\_rid\_tOrders;  SELECT \* FROM ordernum\_ridder;  ALTER EVENT rid\_tOrders DISABLE;  ALTER EVENT rid\_tOrders ENABLE; |