# 需求分析

调研该应用的用户需求，分析未来的扩展用户需求。具体描述每一类用户需求，存储什么数据，怎么存储，数据与需求的关系，谁可以访问这些数据。此部分可参考软件工程的需求分析。

餐饮服务的商家：

接单和处理订单：当用户将订单信息发送给商家，商家接下订单，准备商品；

订单对账：定期核对订单收入与事实是否相符，账目检查等

（送餐区域）

存储的数据：自己商品数据，以往订单数据、

## 消费者

查询：用于查询附近的商品（美食等），可按种类、距离、好评等查询

点单/下单：查询完毕，创建订单，购买商品

查看订单进度：用户可以随时查看订单的进度，当骑手送订单时可以查询到骑手（即订单的位置）

## 骑手

送单：骑手查询需要配送的订单，选择订单进行配送

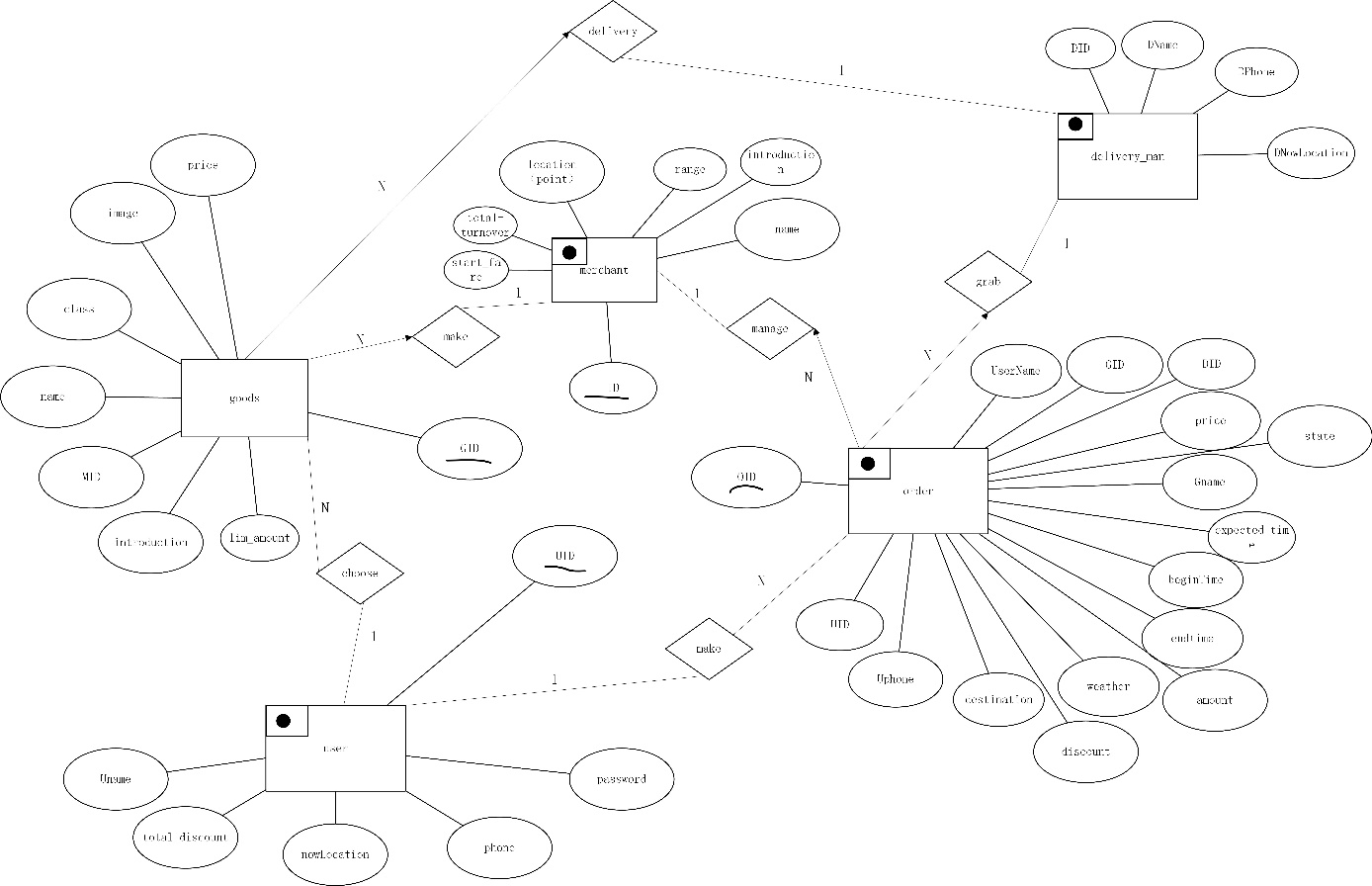
## 其他

评价：用户消费完毕后可以对订单进行评价

注册功能：有新的用户注册时，验证用户ID是否存在，不存在则添加该新用户记录

# 概念设计

本系统用到的涉及的实体有用户、商家、商品、配送员、订单。当用户购买物品、商家准备好物品、配送员把物品配送至用户是创建一个完整的订单。未完成订单也可以保留为不完整的订单。



1. 关系merchant中ID为主键，location为商家的位置点geometry：：point
2. 关系goods中GID为主键，MID为外键依据merchant中的ID
3. 关系user中UID为主键，nowLocation为当前登录的地点geometry：：point
4. password长度必须大于5
5. 关系delivery\_man中DID为主键，DNoWLocation是骑手的当前位置，每隔一定时间刷新一次
6. 关系order中OID为主键，UID是外键依据user中的主键UID，GID为外键依据goods中的的主键GID，DID为外键依据delivery——man中的主键DID，destination为订单的目的地geometry：：point

其中state 0 为未支付，1为已经支付商家未处理，2 为商家已经处理未在配送， 3 为配送中， 4 为配送完成，5 为退货等

# 逻辑设计

## merchant

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 其他 |
| ID | varchar(15) | primary key |
| location | geometry::point | 商家的地址 |
| name | varchar（15） | 商家的名称 |
| range | int | 商家配送公里数，超出无法配送 |
| introduction | varchar（100） | 商家的简单介绍 |
| total\_turnover | double | 商家总营业额 |
| start\_fare | float | 配送起步价 |

函数依赖：ID->location、name、range、introduction 、total\_turnover/statfare符合BC范式

## goods

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 其他 |
| GID | varchar（15） | primary key |
| name | varchar（15） | 商品名称 |
| class | int | 1-主食，2-面食等类型 |
| image | byte | 商品的图片 |
| price | float | 商品价格 |
| introduction | varchar(15) | 商品的价格 |
| MID | varchar（15） | foreign key—merchant |
| lim\_amount | int | 限制购买数量 |

函数依赖GID->name、class、image、price、introduction、MID、lim\_amount 符合BC范式

## user

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 其他 |
| UID | varchar（15） | primary key |
| Uname | varchar（15） | 用户名称 |
| password | varchar（15） | 用户密码 |
| phone | varchar（11） | 用户手机号码 |
| NowLocation | geometry：：point | 用户当前位置 |
| total\_discount | float | 优惠金额 |

函数依赖：UID->Uname、password、phone NowLocation、total\_discount符合BC范式

## delivery\_man

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 其他 |
| DID | varchar（15） | primary key |
| Dname | varchar（15） | 骑手名字 |
| Dphone | varchar（11） | 骑手手机号码 |
| DNowLocation | geometry：：point | 骑手当前位置、实时更新 |

函数依赖 DID->Dname、Dphone、DNowLocation 符合BC范式

## order

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 其他 |
| OID | varchar（20） | primary key |
| UID | varchar（15） | foreign key –user |
| GID | varchar（15） | foreign key –goods |
| DID | varchar（15） | foreign key-Delivery\_man |
| UserName | varchar (15) | 用户名称 |
| price | float | 商品价格 |
| Gname | varchar (15) | 商品名称 |
| expected\_time | time | 期望送达时间 |
| begin\_time | time | 订单开始时间 |
| end\_time | time | 订单结束时间 |
| weather | varchar(15) | 天气 |
| destination | geontry::point | 送货地点 |
| Uphone | varchar(11) | 用户号码 |
| state | int | 订单状态，代表是否支付等 |
| amount | int | goods数量 |
| discout | float | 优惠金额 |

函数依赖：OID->expected\_time 、begintime、endtime、weather、destination 、state、amoun、discout；UID->Username、Uphone、destination； GID->Gname 不符合BC范式 关系的keys：OID、UID、GID；

可分解为

orderInfo（OID,expected\_time、begintim,endtime、weather、destination、state、amoun、discout）、

orderInfo2（OID、GID、UID，DID，MID）

# 物理设计

## 视图创建

商家查询订单：为每个商家创建一个由该商家完成的订单视图

商家增加商品：为每个商家创建一个该商家的所有商品视图

用户查询订单：为每个用户创建一个由该用户完成的订单视图

配送员查询订单：为每个配送员创建一个该配送员完成的订单视图

## 用户权限控制

**综述**：对商家、用户、配送员三类用户都只能修改自己的相关信息，所以其权限分配都对应相应的视图

**商家用户权限**：每一个商家可以选择、更新自己商店的信息，选择、更新、插入自己商店的商品信息，选择商店的订单信息，注册时可以对应表中插入对应的信息

**用户权限**：每一个用户可以选择、更新自己的账号信息，可以选择自己完成的订单信息，在下单时可以插入订单信息，注册时可以对应表中插入对应的信息，用户可以选择商家、商品信息

**配送员权限**：每一个配送员可以选择、更新自己账号信息，可以选择自己完成的订单信息，可以选择未被配送员配送的订单信息，更新未被配送的订单中的配送员DID，更新代表该配送员将配送这一订单，注册时可以对应表中插入对应的信息

**管理员权限**：可以为各种用户授权，选择、更新、插入、删除相应关系

## 触发器创建

1. 每单订单完成后，自动对商家营业额进行累加，正常完成则累加，退款或者取消则不累加

create or replace function updataTriggerorder()

returns trigger as $$

begin

updata merchant set total\_turnovers=(select total\_turnovers from mechant where mid=new.mid)+amount\*price where mid=new.mid

return new;

end;

$$language plpgsql;

drop trigger if exists updatatrigger on morder;

Create trigger updataTrigger

after updata on morder

For Each Row

execute procedure updatatriggerorder();

1. 每单订单完成后自动对用户进行优惠金额叠加

create or replace function updataTriggerorder1()

returns trigger as $$

begin

updata user set total\_discout=(select total\_discout from user where uid=new.uid)+new.discount;

return new;

end;

$$language plpgsql;

drop trigger if exists updatatrigger1 on morder;

Create trigger updatatrigger1

after updata on morder

For Each Row

execute procedure updataTriggerorder1 ();

**3**、每单订单创建时，对商品数量进行检查，如果超过限购范围，则自动设为限购数量

该触发器已在后面对视图的插入中实现

**4**、当各类用户对视图进行修改时，创建触发器，修改基本表

（1） 用户对订单进行插入时

create or replace function InsertTriggerorder()

returns trigger as $$

declare

lim int;

begin

lim=(select lim\_amount from goods where goods.gid=new.gid)

if(lim not null and lim<new.amout) then set new.amout=lim;

insert into order\_info values (new.oid, new.expected\_time,new.begintime, new.endtime,new.weather,new.destination,new.state);

insert into ordr\_infor2 values(new.oid,new.gid,new.did.new.uid,new.mid);

return new;

end;

$$language plpgsql;

drop trigger if exists inserttrigger on morder;

Create trigger InsertTrigger

instead of insert on morder

For Each Row

execute procedure inserttriggerorder();

（2）外卖员接单时

create or replace function updateTrigger()

return trigger as $$

begin

update order\_infor2 set did = new.did where oid=new.oid

return new;

end;

$$language plpgsql;

drop trigger if exisits updatetrigger on order\_no\_delivery

create trigger updatatrigger

instead of insert on order\_no\_delivery

for Each row

execute procedure updatetrigger();

# 数据库实施

## 1、关系创建语句

**创建商家关系**

CREATE TABLE merchant(

ID varchar(15) primary key,

location geometry,

name varchar(15),

range int,

introduction varchar(100),

total\_turnover double,

start\_fare float

)

**创建商品关系**

CREATE TABLE goods(

GID varchar(15) primary key,

name varchar(15),

class int,

image bytea,

introduction varchar(100),

MID varchar(15) REFERENCES merchant(ID),

lim\_amout int

)

**创建用户关系**

CREATE TABLE user1(

UID varchar(15) primary key,

Uname varchar(15),

password varchar(15),

phone varchar(11),

nowLocation geometry,

total\_discount

)

**创建配送员关系**

CREATE TABLE delivery\_man(

DID varchar(15) primary key,

Dname varchar(15),

Dphone varchar(11),

NowLocation geomytry

)

**创建订单关系**

CREATE TABLE order\_info(

OID varchar(20) primary key,

expected\_time time,

begintime time,

endtime time,

weather varchar(15),

destination geometry，

state int,

amount int ,

discount float,

)

CREATE TABLE order\_info2(

OID varchar(20) REFERENCES order\_info(OID),

GID varchar(15) REFERENCES GOODS(gid),

DID varchar(15) REFERENCES delivery\_man(did)

MID varchar（15）REFERENCES merchant（ID），

UID varchar（15）REFERENCE User（UID）

)

## 2、视图创建

CREATE or REPLACE VIEW morder

as (SELECT order\_info2.OID,order\_info2.UID,order\_info2.GID,order\_info2.DID, order\_info2.MID，user1.uname as usernmae,price,aoumt,discount,price\*amout as total\_cost

goods.name as goodsname,expected\_time,begintime,endtime,weather,destination,user1.phone as upone

,state

from order\_info,order\_info2,user1,merchant,goods,delivery\_man

where order\_info.oid=order\_info2.oid

and order\_info2.gid=goods.gid

and order\_info2.uid=user1.uid

and order\_info2.did=delivery\_man.did

and order\_info2.mid=merchant.id

)

**创建未配送订单视图**

CREATE OR REPLACE VIEW un\_delyver\_order

as (SELECT \* FROM morder where DID='0')

**为单一用户创建订单视图**

CREATE OR REPLACE VIEW USER\_ORDER1

as

select \* from morder

where uid='1'

**为单一商家创建订单视图**

CREATE OR REPLACE VIEW MERCHANT\_ORDER3

as

select \* from morder

where mid='3'

**为单一配送员创建订单视图**

CREATE OR REPLACE VIEW DELIVERYMAN\_ORDER1

as

select \* from morder

where did='1'

## 3、索引创建

为商家地址创建索引

CREATE INDEX merchant\_l on merchant using Gist(location)

## 4、用户授权

**给配送员授权查看未配送订单**

GRANT SELECT(\*),UPDATE(DID) ON un\_delyver\_order

TO DELIVERY\_MAN1

**给用户授权查看自己的订单**

GRANT SELECT ON user\_order1

TO user1

## 5、数据插入语句

**用户下单插入订单数据**

insert into morder values（‘1’，‘1’，‘1’，‘1’，‘小明’，15,2,30，‘酸菜鱼’，null，null，null，‘sunny’，（select nowlocation from user where uid=’1’）,’158774656547’,1,）

## 6、用需求查询

1. **配送员查询没有被配送的订单**

SELECT \* FROM morder WHERE DID=’0’ and state=1

did等于0时代表此单没有配送员，state=1代表订单已经支付

1. **用户查询附近的美食**

select id,location,merchant.name,gid,goods.name,price from merchant,goods,user1

where goods.mid=merchant.id and

ST\_dwithin(geography(merchant.location),geography(user1.nowlocation),20000)

and uid='1'

1. **用户查询配送进度**

select destination,nowlocation from

USER\_ORDER1,delivery\_man

where USER\_ORDER1.did=delivery\_man.did

and uid='1' and state=3

1. **商家接单，准备商品**

select \* from merchant\_order3

where state=1