浙江大学

课程: 地理空间数据库

题目: LBS 空间数据库设计——饿了么

姓名: 张思远，秦卫付

学号: 3170103112，3170106249

专业班级: 地理信息科学 1701

教师: 陶煜波

目 录

一、 引言

1. 研究背景

“饿了么”是现今中国最大的餐饮 O2O 平台，在近十年的时间里，它极大地推动了中国餐饮行业的数字化进程，为人们的饮食带来了极大的便利与更为多样化的选择。而作为大学生群体，饿了么在线外卖平台也逐渐成为我们除堂食以外的一个不可或缺的常规就餐方式。

本次实习就以当下热门的外卖 APP“饿了么”为研究对象，调研其用户需求与服务器端的数据库设计，借此进一步加深对数据库设计流程的理解。

应用名称：饿了么 操作系统：Android/IOS 软件版本：8.7.1 更新时间：2019/5/20

图 1 饿了么客户端架构

2. 实习目的

本次实习选取当下热门的在线外卖 APP“饿了么”作为研究对象，调研其用户需求，分析其服务器端数据库架构与设计。从开发者的角度，自源头做起，切身体会空间数据库的设计流程与关键要点，也借此进一步理解、回顾所学的 E/R 图、约束条件、关系、函数依赖、索引、视图、触发器、用户权限及数据插入查询等课程知识，能够熟练地应用于具体问题的解决。

二、 需求分析

需求分析是数据库设计的指导思想和根本依据，在本次实习中，本着面向对象的设计理念，先分析用户的具体功能需求，在具体剖析“饿了么”的主要任务与功能模块。本节共分为用户需求分析、任务、数据流图及数据字典四部分，将具体阐述每一类用户需求，及相应的存储数据种类，存储方法及访问权限。

1. 用户需求分析

1.1 客户需求分析

点餐客户进入 APP，首先打开主页面，进入“美食”、“卖场便利”等进行浏览，但在选购完商品后系统会提示注册/登陆，在验证用户的合法性后，创建订单信息。订餐用户可以依据地理位置和商圈查询附近所有的餐厅，和餐厅推出的菜品和价格，也可依据口味、菜系、时段等进行查询，并可对完成后的消费服务进行评价；

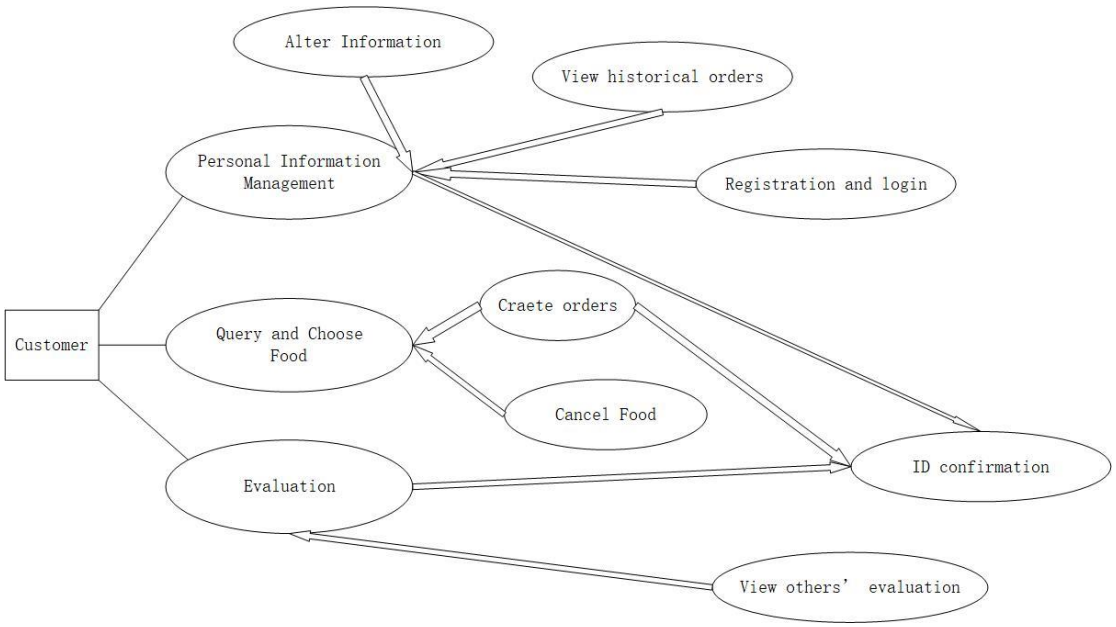

图 2 客户模块需求简明

1.2 商家需求分析

商家管理者将对系统内容进行管理，添加新的产品、查询已提交的订单、管理店铺信息、管理“满减”活动、回复客户评价等等；

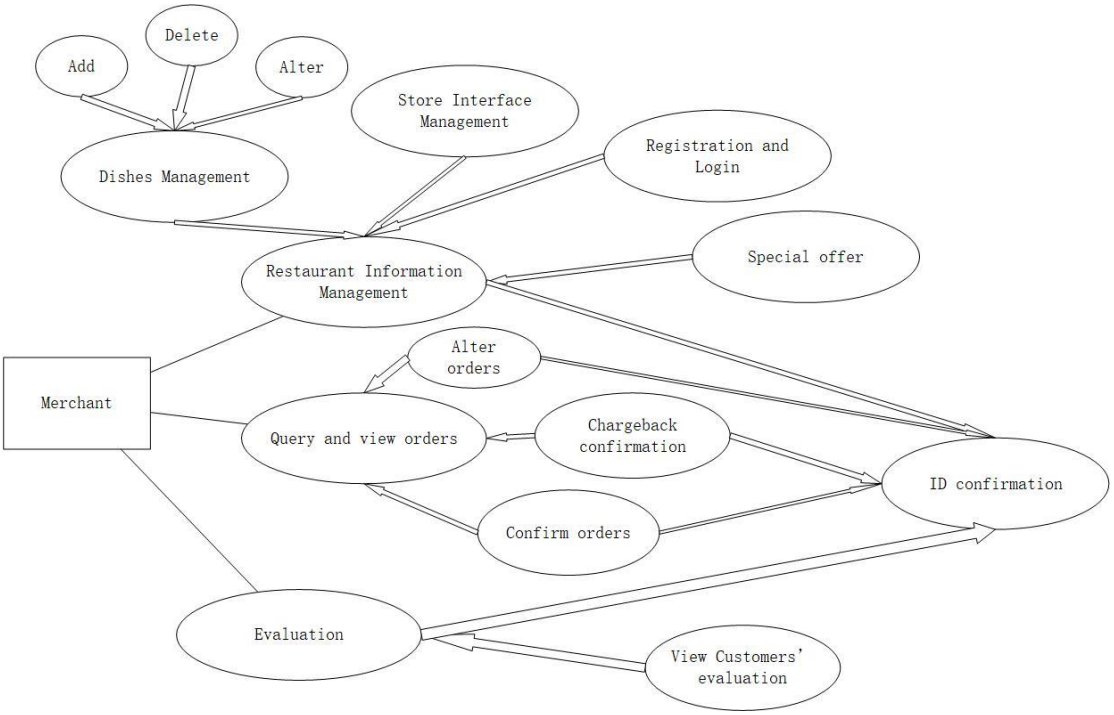

图 3 商家模块需求简明

1.3 管理员需求分析

系统管理员能够对系统内部的所有用户的权限以及用户信息进行管理，例如分配商家管理者等用户的操作权限。负责 APP 页面的跳转与数据库数据的修改，用户信息管理与查询服务，发布通知等等，但不参与经营管理。

2. 任务

- 用户登录：设置用户登录界面，该处用户能够通过既定的用户名称和登录密码，然后通过对录入用户信息的验证，判断是否将用户引入主界面。登录成功，则将登录信息存储至连接线程中，并为后续操作提供访问接口。
- 用户注册：该功能主要为**游客**提供注册功能，其中包括录入信息的界面，检测用户所填入注册数据的合法性，并对应数据库的表中保存注册用户所填信息。**该功能只对系统管理员开放**。
- 注册用户信息管理：该功能是对用户信息进行管理。**管理员可以通过该模块对用户信息进行更改、删除等操作**，同时可以定义权限分配。**注册用户可以通过该模块对个人的信息做一定的修改**。

- 订餐人员信息查阅：对**已进入系统人员**提供菜品信息的查询功能，添加商品、结算、更改密码、红包兑换及订单评价等。对**非注册用户**提供菜品信息的查询和浏览功能。
- 餐厅管理：为**商家管理者**提供对餐厅信息的管理。其主要功能有餐厅详细信息的增

加、删除和修改等操作。包含餐厅编号、名称、所在区域、送餐起步价、联系电话等。

- 商圈管理：查看、修改、添加商圈子类、删除商圈信息，并及时更新商圈信息在页面上显示。 **该权限只对系统管理员开放。**
- 菜品管理：为**餐厅管理员**提供对菜品的增加、移除、升级维护这些功能。菜品信息主要有菜品的编号、具体名称、售价、实拍、以及其他更详细的店铺相关的信息。
- 活动公告：该模块是一个活动发布的渠道。**系统管理员**能够通过该模块对活动信息和相关公告进行管理操作。内容主要有活动的 ID、标题、具体的内容和浏览用户的评论信息等等与活动相关的信息。

3. 数据流图

数据流动的路径和方向通常采用数据流图来表示。数据流图从数据产生到流动到加工再到输出，使用结构化需求分析方法，以“自顶向下，逐层分析”的思想对 APP 的数据存储与流动过程进行剖析。

3.1 顶层数据流图

在饿了么外卖在线订餐平台中，其外部用户主要有客户、商家、配送员和系统管理员。其中，客户享有订购菜品及订单和书籍等信息查询的功能，商家享有管理菜品及订单和菜品查询功能，配送员享有获取配送信息等功能，管理员可对系统的各种信息进行管理和维护。

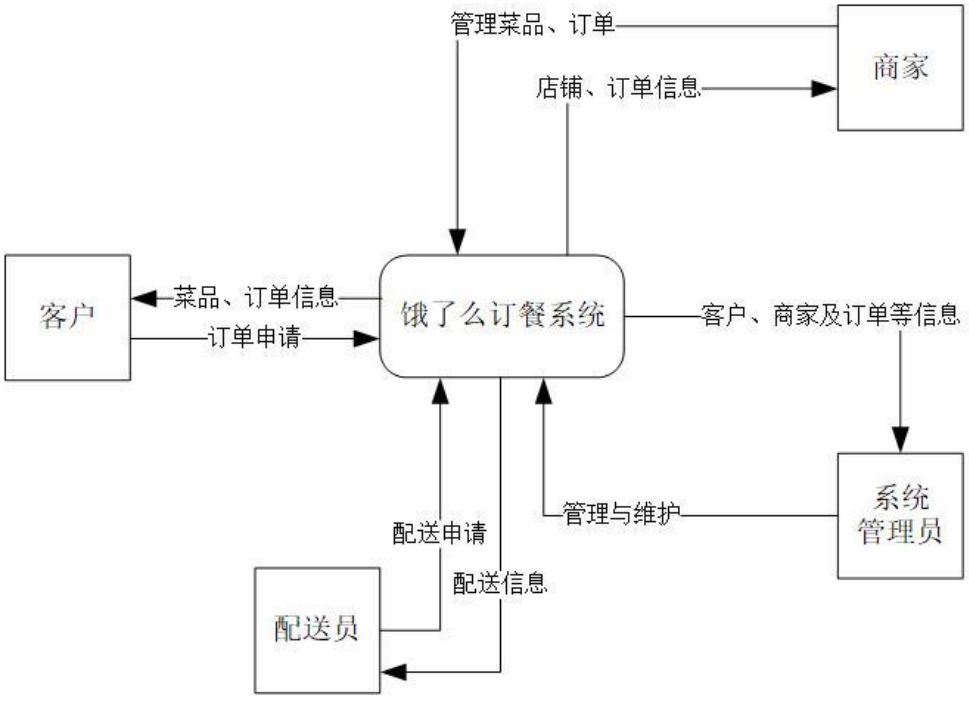

图 4 顶层数据流图

3.2 0层数据流图

在0层数据流图中，具体列出了客户享有的订购菜品及菜品、订单查询功能，商家享有的菜品信息管理与订单查询功能，配送员的抢单功能。

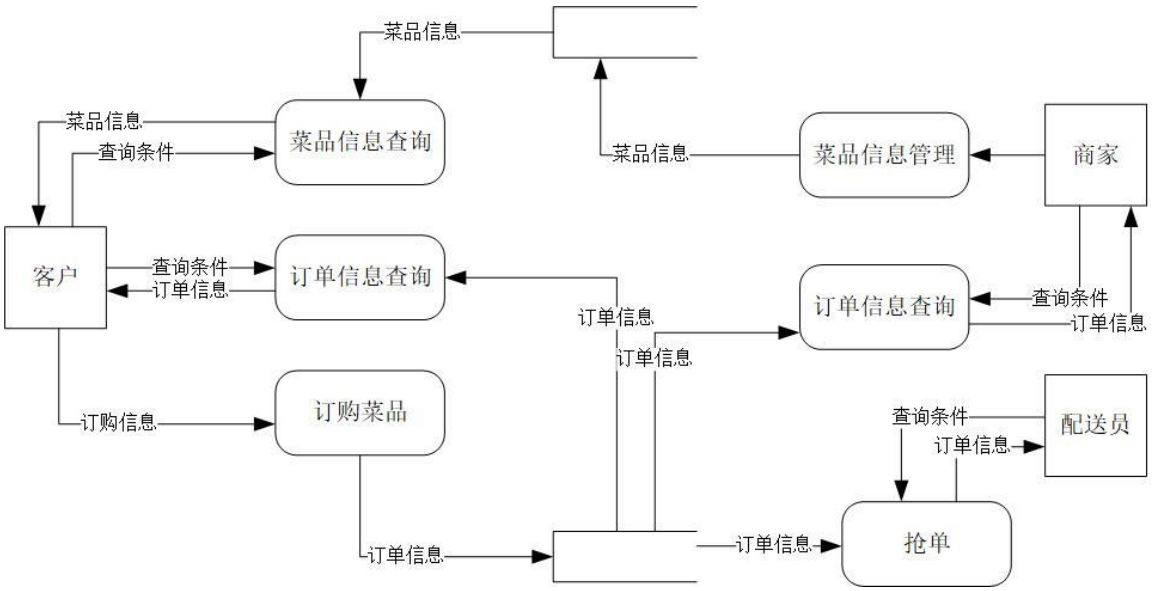

图5 0层数据流图

三、 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；
(7) 其中state对应以下几种状态：0-未支付，1-客户已支付，商家未确认订单，2-商家已确认订单，配送员未取货， 3-配送中，4-配送已到达，订单完成，5-退货等。

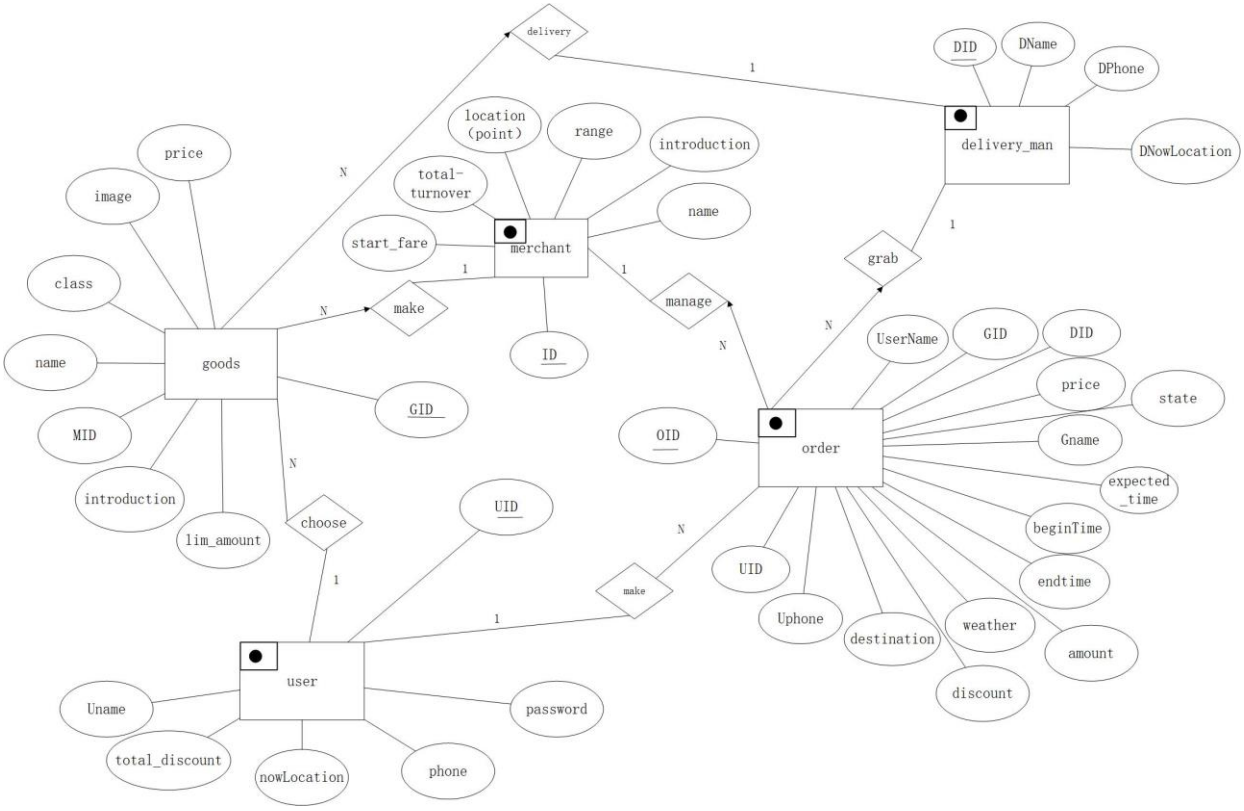

图 6 “饿了么”在线外卖系统简化 E-R 图

四、 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	geometry::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
Discount	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）

五、 物理设计

5.1 性能要求分析

1、 响应时间：响应时间是用户感受软件系统为其服务所耗费的时间，由服务器端、网络及客户端响应时间组成，经手机 APP 测试，“饿了么”在网络良好情况下，每次点击响应时间在 1-2 秒；

2、 吞吐量：在本实习中，“吞吐量”特指数据库 SQL 语句在单位时间内的执行数量。“饿了么”现有用户超 2.6 亿，高峰期下单数以百万计，需要系统较高的查询效率，合理的数据存储方式。

3、 可靠性与精度：对于查询来说，所有在相应域中包含查询关键字的记录都应能查到，同时保证准确率。“饿了么”可以直接通过关键字段精确查询，也可通过商圈、菜品类别等进行分类检索。

4、 可移植性：可从某一环境转移到另一环境的能力。适应性、易安装性、一致性（遵循性）、可替换性。“饿了么”能很好地适应 ios，android，c 端的与运行。

5.2 关系创建

关系创建语句：

1、 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

)
```

2、 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
)
```

3、 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
)
```

4、 创建配送员关系

```
CREATE TABLE delivery_man(
        DID varchar(15) primary key,
        Dname varchar(15),
        Dphone varchar(11),
        NowLocation geomytry
)
```

5、 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）
)
```

5.3 索引创建

1、 在用餐高峰期，商家与配送员会频繁查询、确认、检查自己所负责的订单。为应对这一高数量访问，在 order 表的 MID,DID 属性上建立索引在商家及配送员查询订单时，只需查询自己所在店铺/自己负责配送的订单。

- 商家： CREATE INDEX Merorder on order(MID ASC)
- 配送员： CREATE INDEX Delorder on order(DID ASC)

2、 为商家地址创建索引。

```
CREATE INDEX merchant_l on merchant using Gist(location)
```

3、 订餐客户在查询商家所提供的菜品时，是根据食品名称进行查询的，故可在 goods 表的 name 属性上建立聚簇索引，以应对高峰期频繁的查询。

```
CREATE CLUSTER INDEX Foodname on goods(name ASC)
```

5.4 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

```
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,
 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'
```

5.5 触发器创建

- 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();
```

- 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 ();
```

- 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();
```

- 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*已在之前对视图的插入中实现*。

5.6 用户权限分析

5.6.1 **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

5.6.2 **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

5.6.3 **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

5.6.4 **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

5.6.5 **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

举例：

- **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
```

六、 数据库实施

6.1 数据插入

6.1.1 系统管理员权限

- 商家插入

```
INSERT INTO merchant values(
```

```
1,ST_GeometryFromTExt('point(120.113688 30.305505)',4326),
'张小哥在杭州',5,'串串搞起来')
INSERT INTO merchant values(
2,ST_GeometryFromTExt('point(120.100249 30.320657)',4326),
'阿秦米线',5,'百年老店')
```

- 菜品插入

```
INSERT INTO goods(GID,name,class,price,introduction,mid)
 values(1,'烤鱼',1,15,'好吃不贵',2)
```

- 配送员插入

```
INSERT INTO delivery_man values(
3，’小丽’,'15287814721',
ST_geometryFromText('point(120.076331797721 30.3043168)',4326))
```

- 订单插入

```
INSERT INTO order_info(oid ,begintime,weather,destination)
 values(1,(select now()),'sunny',
(select nowlocation from user1 where uid='1'))

INSERT INTO order_info2 values(1,1,1，1)
```

6.1.2 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,）

6.2 数据查询

6.2.1 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

6.2.2 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'
```

6.2.3 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
```

6.2.4 商家接单，准备商品

```
select * from merchant_order3
where state=1
```