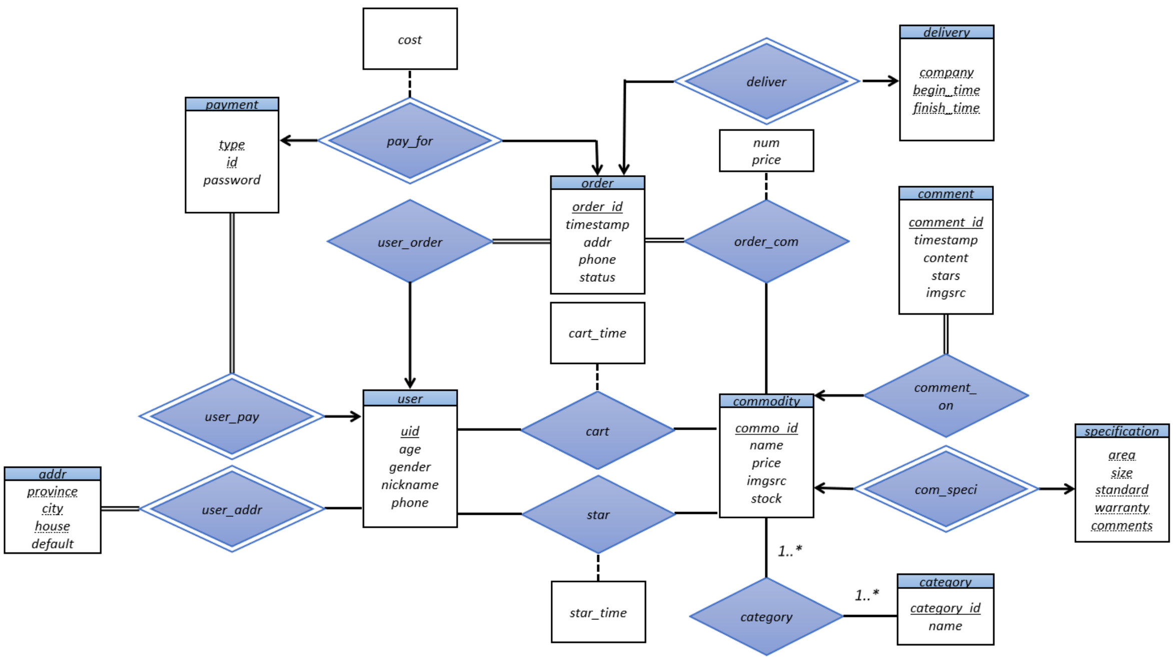
1.



> *id* of *payment* refers to the id of the payment, like the id of a credit card.
>
> *addr* of *order* is in fact *province*, *city* and *house*. I use *addr* for convenience.

2.

user(<u>uid</u>, age, gender, nickname, phone)

addr(<u>uid</u>, <u>province</u>, <u>city</u>, <u>house</u>, default)

payment(<u>uid</u>, <u>type</u>, <u>id</u>, password)

order(<u>order_id</u>, uid, timestamp, addr, phone, status, company, begin_time, finish_time, type, id, cost)

order_com(<u>order_id</u>, <u>commo_id</u>, num, price)

cart(<u>uid</u>, <u>commo_id</u>, cart_time)

star(<u>uid</u>, <u>commo_id</u>, star_time)

commodity(<u>commo_id</u>, name, price, imgsrc, stock, area, size, standard, warranty, comments)

cate(<u>commo_id</u>, <u>category_id</u>)

category(<u>category_id</u>, name)

comment(<u>comment_id</u>, timestamp, content, stars, imgsrc, commo_id)

> *commo_id* of *order* may be multivalued or needs an extra table when an order includes multiple commodities

3.

```
create table user(
```

```sql
    uid char(20),
    age int,
    gender char(1),
    nickname varchar(20),
    phone int,
    primary key (uid),
    check (gender in ('F', 'M')));

create table addr(
    uid char(20),
    province varchar(20),
    city varchar(20),
    house varchar(20),
    default bool,
    primary key (uid, province, city, house),
    foreign key uid references user(uid));

create table payment(
    uid char(20),
    type int,
    id char(20),
    password varchar(20),
    primary key (uid, type, id),
    foreign key uid references user(uid));

create table category(
    category_id int,
    name varchar(20),
    primary key (category_id));

create table commodity(
    commo_id char(20),
    name varchar(30),
    price float,
    imgsrc varchar(50),
    stock int,
    area varchar(30),
    size varchar(30),
    standard varchar(10),
    warranty char(5),
    comments int,
    primary key (commo_id),
    foreign key category_id references category(category_id));

create table cate(
    category_id int,
    commo_id char(20),
    primary key (category_id, commo_id),
    foreign key category_id references category(category_id),
    foreign key commo_id references commodity(commo_id));

create table order(
    order_id char(20),
    uid char(20),
    timestamp timestamp,
    province varchar(20),
    city varchar(20),
    house varchar(20),
    phone int,
    status char(1),
    company char(4),
    begin_time int,
    finish_time int,
    type int,
    id char(20),
    cost float,
    primary key (order_id),
    foreign key (uid, type, id) references payment(uid, type, id),
    foreign key (commo_id) references commodity(commo_id),
    check (status in ('P', 'D', 'F')));
# P means payed
# D means delivering
# F means finished

create table order_com(
    order_id char(20),
    commo_id char(20),
    num int,
    price float,
    primary key (order_id, commo_id),
    foreign key order_id references order(order_id),
    foreign key commo_id references commodity(commo_id));

create table comment(
    comment_id char(20),
    timestamp timestamp,
    content varchar(200),
    imgsrc varchar(50),
    stars int,
    commo_id char(20),
    primary key (comment_id),
    foreign key commo_id references commodity(commo_id));

create table cart(
    uid char(20),
    commo_id char(20),
    cart_time timestamp,
    primary key (uid, commo_id),
    foreign key uid references user(uid),
    foreign key commo_id references commodity(commo_id));

create table star(
    uid char(20),
    commo_id char(20),
    star_time timestamp,
    primary key (uid, commo_id),
    foreign key uid references user(uid),
    foreign key commo_id references commodity(commo_id));
```

4.

1)

```
select order_id, MAX(total)
from (select order_id, count(*) as total
     from order_com
     group by order_id)
group by order_id;
```

2)

```
select uid, order_id
from order
where status = 'P';
# P means payed
# if the commodity is delivered, ths status is D
```

3)

```
select order_id
from order
where uid = "DBS" and province = "ZheJiang" and city = "Hangzhou" and house =
"ZJG";
```

4)

```
select commo_id
from (cate natural join category) as A
where A.name = "儿童" and exists (select *
                        from (cate natural join category) as B
                        where A.commo_id = B.como_id and B.name = "图书");
# or making use of bit operation or multivalued attribute will be easier
```

5)

```
select commo_id, COUNT(num) as sales
from order natural join order_com
where timestamp between TIMESTAMP('2019-01-01 00:00:00') and TIMESTAMP('2020-01-
01 00:00:00')
group by commo_id
order by sales desc
limit 10;
```

6)

```
begin;

select sales into @a
```

```
from commodity
where commo_id = "p1_id";
update commodity
set stock = @a - num1,
where commo_id = "p1_id";
select sales into @a
from commodity
where commo_id = "p2_id";
update commodity
set stock = @a - num2
where commo_id = "p2_id";

select province, city, house into @p, @c, @h
from addr
where uid = "c1_id" and default = TRUE;
set @cost = num1 * price1 + num2 * price2;
# assume that the phone number, type and id of the payment are given
# the information of delivery needs to be updated afterwards
insert into order("o1_id", "c1_id", CURRENT_TIMESTAMP(), @p, @c, @h, @phone,
'P', NULL, NULL, NULL, @type, @id, @cost);
insert into order_com values ("o1_id", "p1_id", num1, price1);
insert into order_com values ("o1_id", "p2_id", num2, price2);

commit;
```