【MySQL】ER模型(十六)

> 🚚MySQL学习·第十六站~
> 🚩本文已收录至专栏：MySQL通关路
> ❤️‍🩹文末附全文思维导图，感谢各位点赞收藏支持~
> ⭐学习汇总贴，超详细思维导图：【MySQL】学习汇总(完整思维导图)

一.引入

数据库设计是牵一发而动全身的，如果数据库中表非常多，一处有误可能将导致处处需要修改。那有没有什么办法可以提前或者快速看到数据库的全貌呢?比如需要哪些数据表、数据表中应该有哪些字段，数据表与数据表之间有什么关系、通过什么字段进行连接等等。这样我们才能更好的进行整体的梳理和设计。
ER模型就是一个这样的工具。**ER（entity-relationship，实体-联系）模型也叫作实体关系模型，是用来描述现实生活中客观存在的事物、事物的属性，以及事物之间关系的一种数据模型**。在开发基于数据库的信息系统的设计阶段，通常使用ER模型来描述信息需求和信息特性，帮助我们理清业务逻辑，从而设计出优秀的数据库。

二.三要素

ER模型中有三个要素：

- 实体：可以看做是数据对象，往往对应于现实生活中的真实存在的个体，例如用户、商品。在ER模型中，用矩形来表示。实体分为两类，分别是强实体和弱实体。强实体是指不依赖于其他实体的实体;弱实体是指对另一个实体有很强的依赖关系的实体。
- 属性：实体的特性。比如用户的性别、联系电话等。在ER模型中用椭圆形来表示。
- 关系：实体之间的联系。比如一名学生可以学习多门课程，就是一种学生与课程之间的联系。在ER模型中用菱形来表示。

通常，**一个实体集**（class）对应于数据库中的**一个表**(table)，**一个实体**(instance)则对应于数据库表中的**一行**(row)，也称为一条记录（record)。**一个属性**(attribute)对应于数据库表中的**一列(**column)，也称为一个字段(field)。


```
ORM思想 (Object Relational Mapping)体现:
数据库中的一个表  <---> Java中的一个类
表中的一条数据  <---> 类中的一个对象（或实体）
表中的一个列  <----> 类中的一个字段、属性(field)
```

三.关联关系

现实世界中的各种实体之间可能会存在着各种关联联系，对应到数据库中则表现为表与表之间的数据记录可能存在关系(relationship)关系，例如一名学生可以选修多门课程，一门课程可以被多名学生选择。**关联关系可以分为四种类型：一对一关联、一对多关联、多对多关联、自我引用**。

(1) 一对一关联

一对一关联指的是**实体之间的关系是一一对应的**，比如个人与身份证信息之间的关系就是一对一的关系。一个人只能有一个身份证信息，一个身份证信息也只属于一个人。在实际的开发中应用不多，因为一对一可以创建成一张表，通常只用在拆分多字段数据表，减少IO次数。

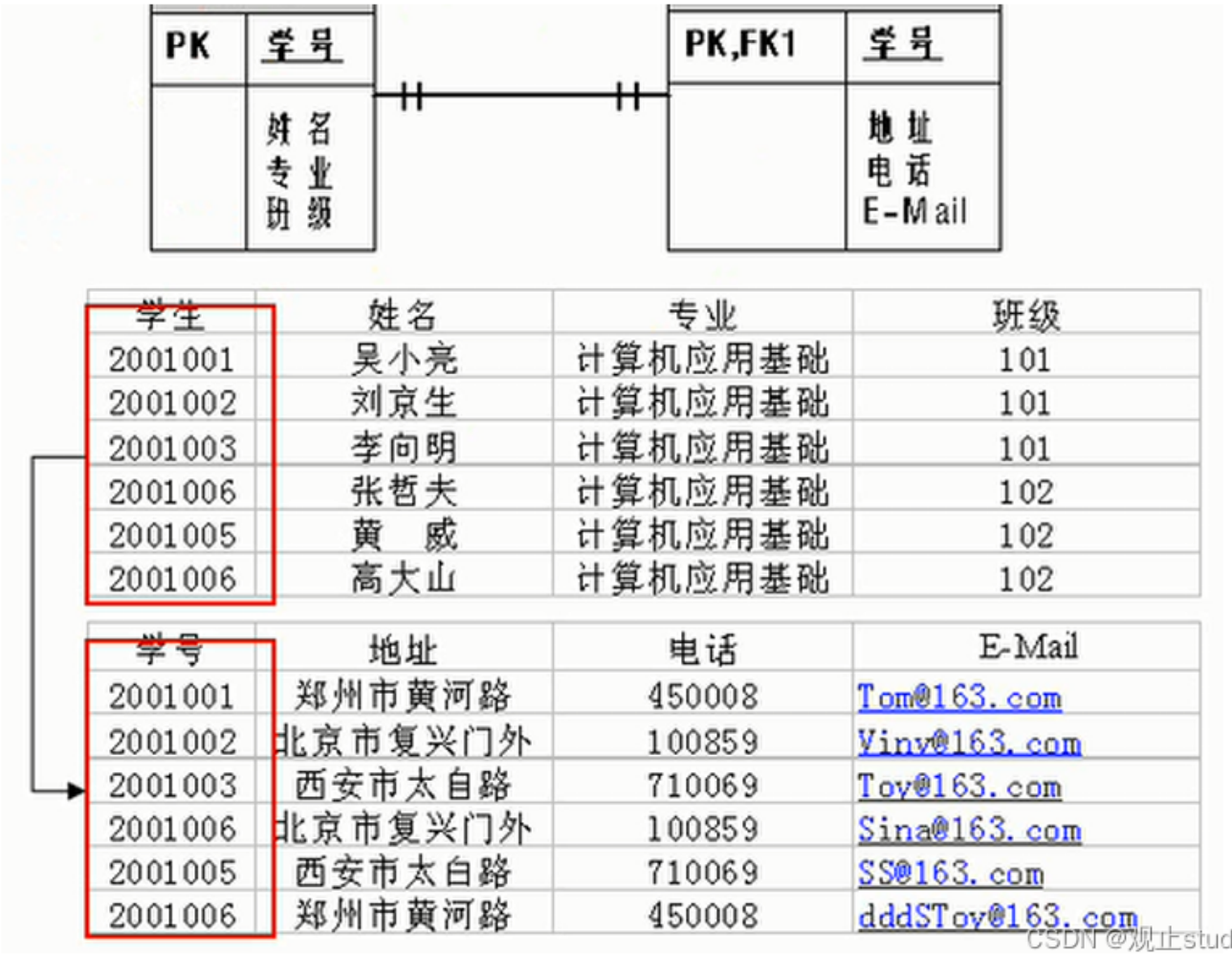
举例:

设计学生表:学号、姓名、手机号码、班级、系别、身份证号码、家庭住址、籍贯、紧急联系人、...
根据字段信息使用频率可以拆为两个表，两个表的记录是一一对应关系：

- 基础信息表(常用信息)：学号、姓名、手机号码、班级、系别
- 档案信息表(不常用信息)：学号、身份证号码、家庭住址、籍贯、紧急联系人、....

两种建表原则：

- 外键唯一：主表的主键和从表的外键（唯一），形成主外键关系，外键唯一。
- 外键是主键：主表的主键和从表的主键，形成主外键关系



学生	姓名	专业	班级
2001001	吴小亮	计算机应用基础	101
2001002	刘京生	计算机应用基础	101
2001003	李向明	计算机应用基础	101
2001006	张哲夫	计算机应用基础	102
2001005	黄　威	计算机应用基础	102
2001006	高大山	计算机应用基础	102

学号	地址	电话	E-Mail
2001001	郑州市黄河路	450008	Tom@163.com
2001002	北京市复兴门外	100859	Vinv@163.com
2001003	西安市太白路	710069	Tov@163.com
2001006	北京市复兴门外	100859	Sina@163.com
2001005	西安市太白路	710069	SS@163.com
2001006	郑州市黄河路	450008	dddSTov@163.com

(2) 一对多关联

一对多关联指的是**一边的实体通过关系，可以对应多个另外一边的实体。相反，另外一边的实体通过这个关系，则只能对应唯一的一边的实体**。比如一个班级有多个学生，而每个学生则只对应一个班级，其中班级与学生就是一对多的关系。

举例：

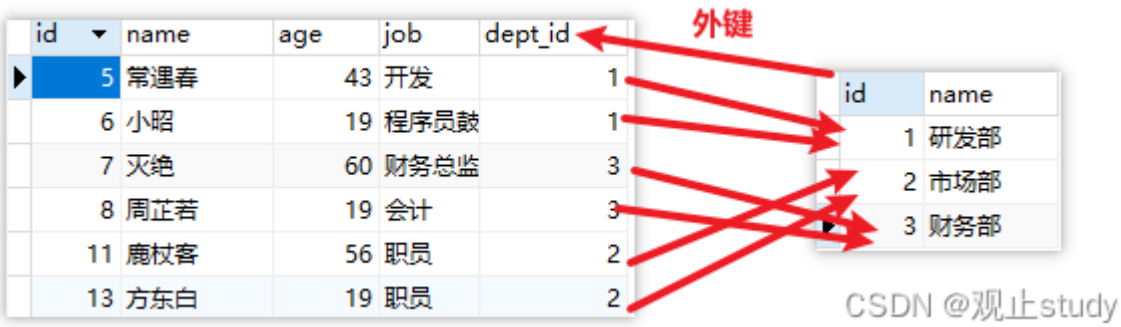
对于学生与课程关系，一名员工只就职于一个部门，而一个部门有多名员工就职。

- 员工表：编号、姓名、...、所属部门。

- 部门表：编号、名称、简介

建表原则:

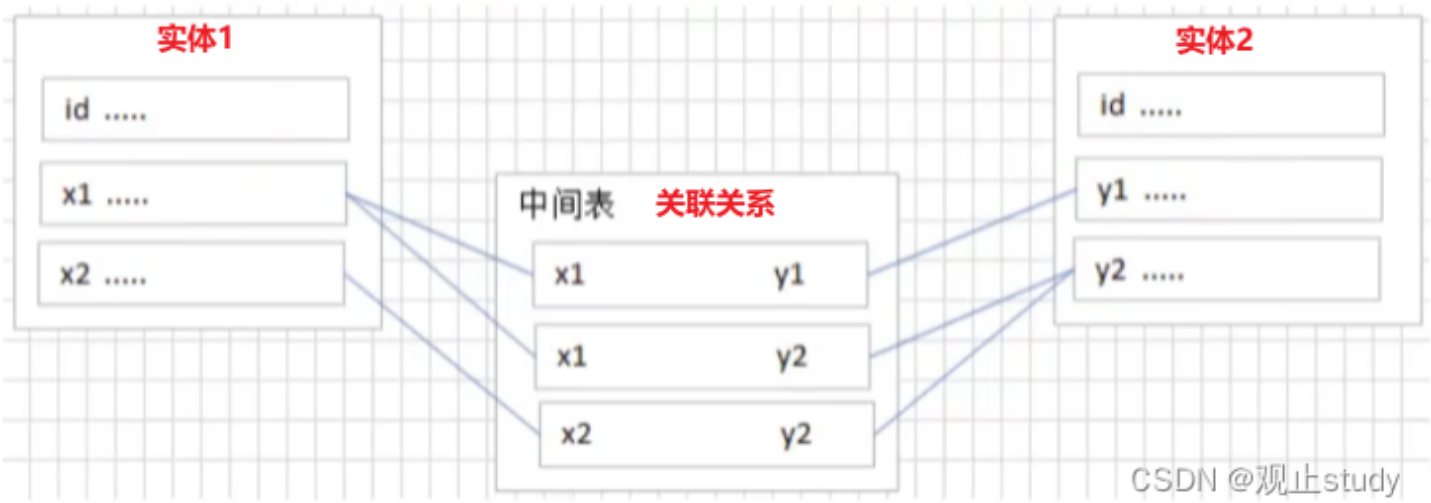
- 在从表(多方)创建一个字段，字段作为外键指向主表(一方)的主键



(3) 多对多关联

多对多关联指的是**关系两边的实体都可以通过关系对应多个对方的实体**。比如一个供货商可以给多个超市供货，一个超市也可以从多个供货商那里采购商品；一门课程可以有多名学生选修，而一名学生也可以选修多门课程，这就是多对多的关系。

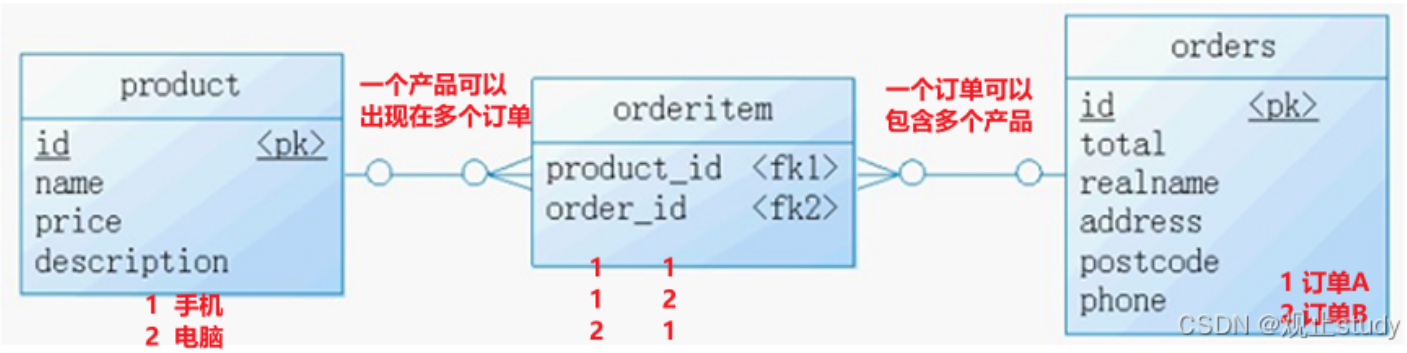
要想表示多对多关系，必须创建第三个表，该表通常称为联接表，它将多对多关系划分为两个一对多关系。将这两个表的主键都插入到第三个表中。



示例:

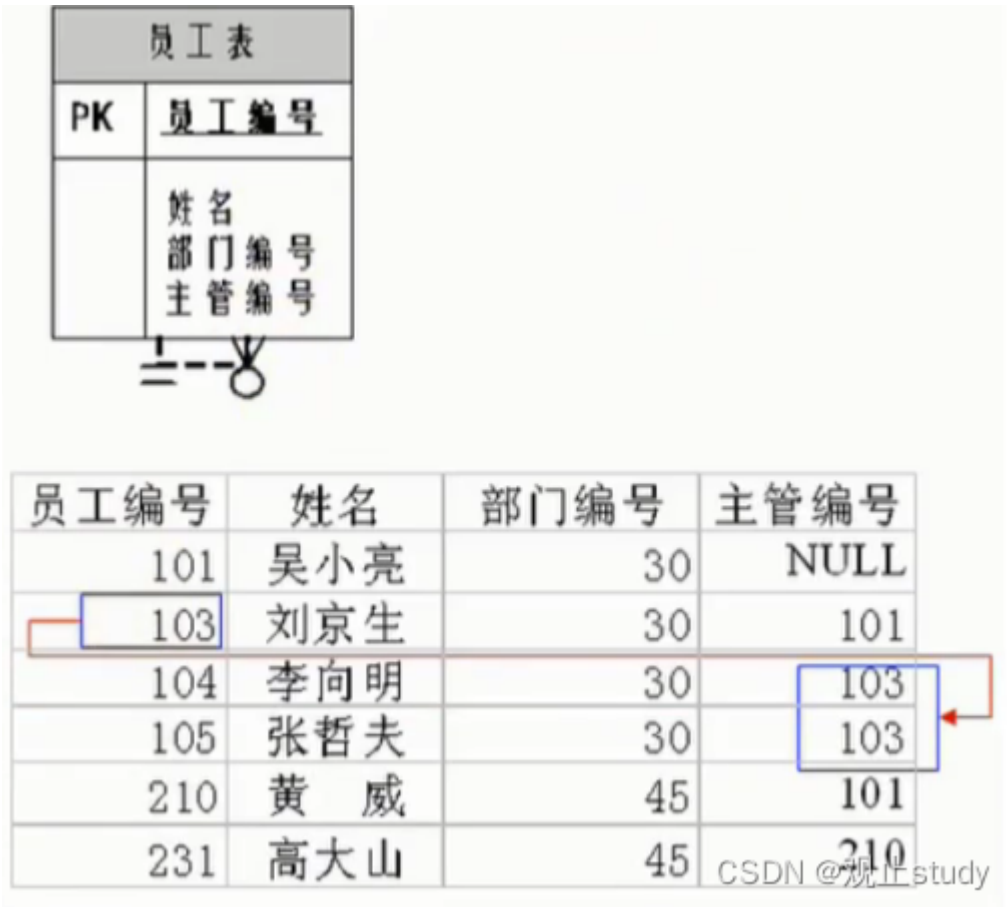
“订单”表和“产品”表有一种多对多的关系，这种关系是通过与“订单明细”表建立两个一对多关系来定义的。一个订单可以有多个产品，每个产品可以出现在多个订单中。

- 产品表：“产品”表中的每条记录表示一个产品。
- 订单表：“订单”表中的每条记录表示一个订单。
- 订单明细表：每个产品可以与“订单”表中的多条记录对应，即出现在多个订单中。一个订单可以与“产品”表中的多条记录对应，即包含多个产品。



(4) 自我引用

自我引用指的是自己引用自己，例如在员工信息表中，A为B的领导，但A也是一个员工。



员工编号	姓名	部门编号	主管编号
101	吴小亮	30	NULL
103	刘京生	30	101
104	李向明	30	103
105	张哲夫	30	103
210	黄　威	45	101
231	高大山	45	210

四.建模分析示例

(1) 引入

ER 模型看起来比较麻烦，但是对于我们把控项目整体非常重要。如果只是开发一个小应用，或许简单设计几个表够用了，不建立ER模型也无所谓，一旦要设计有一定规模的应用，在项目的初始阶段，建立完整的 ER 模型就非常关键 了。**开发应用项目的实质，其实就是 建模**。

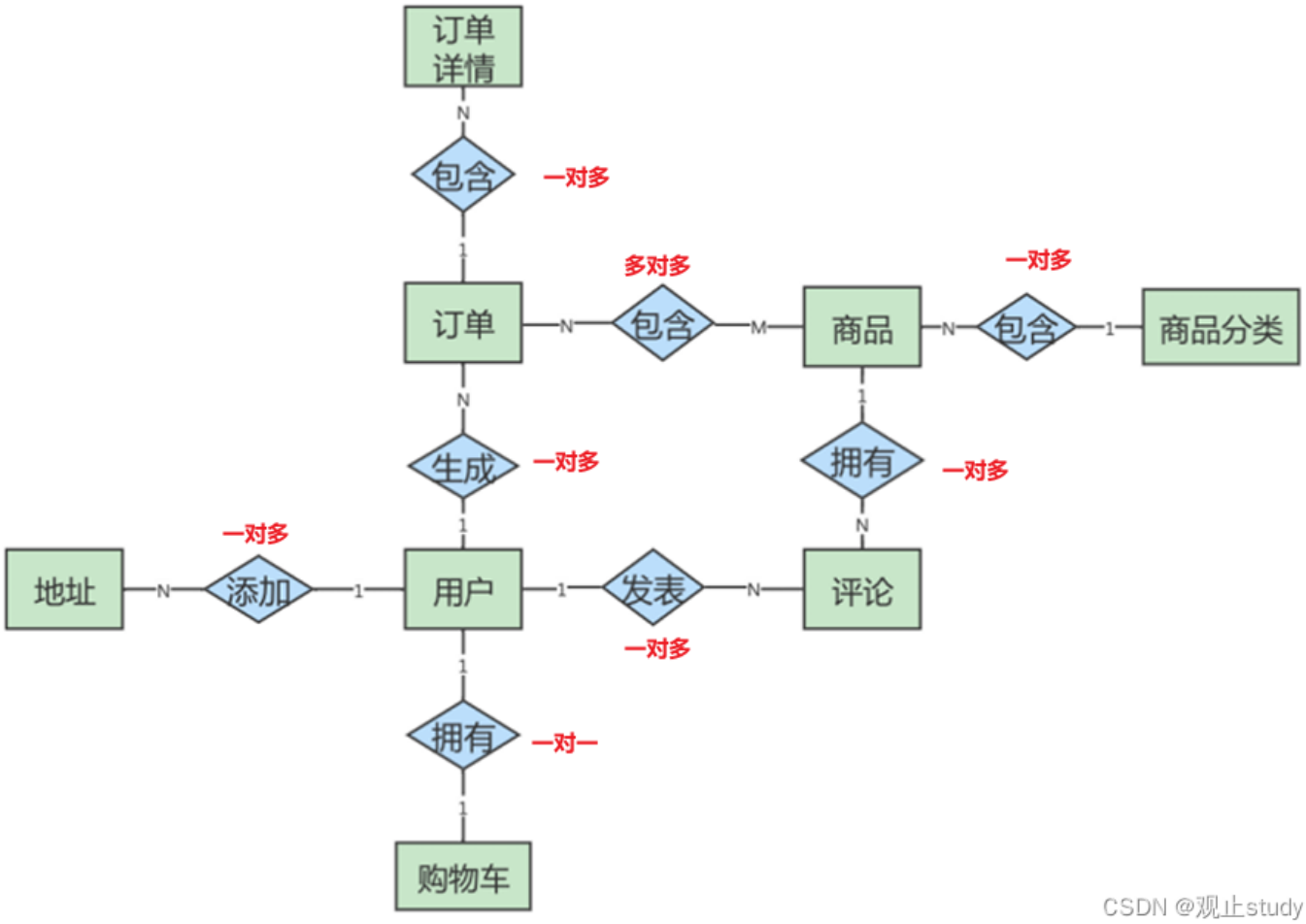
接下来我们来一步一步地针对简化版的电商业务进行建模分析。有如下8个实体：

- 地址实体
- 用户实体
- 购物车实体
- 评论实体
- 商品实体
- 商品分类实体
- 订单实体
- 订单详情实体

其中，用户 和 商品分类是强实体，因为它们不需要依赖其他任何实体。而其他属于弱实体，因为它们虽然都可以独立存在，但是它们都依赖用户这个实体，因此都是弱实体。

(2) 关系分析

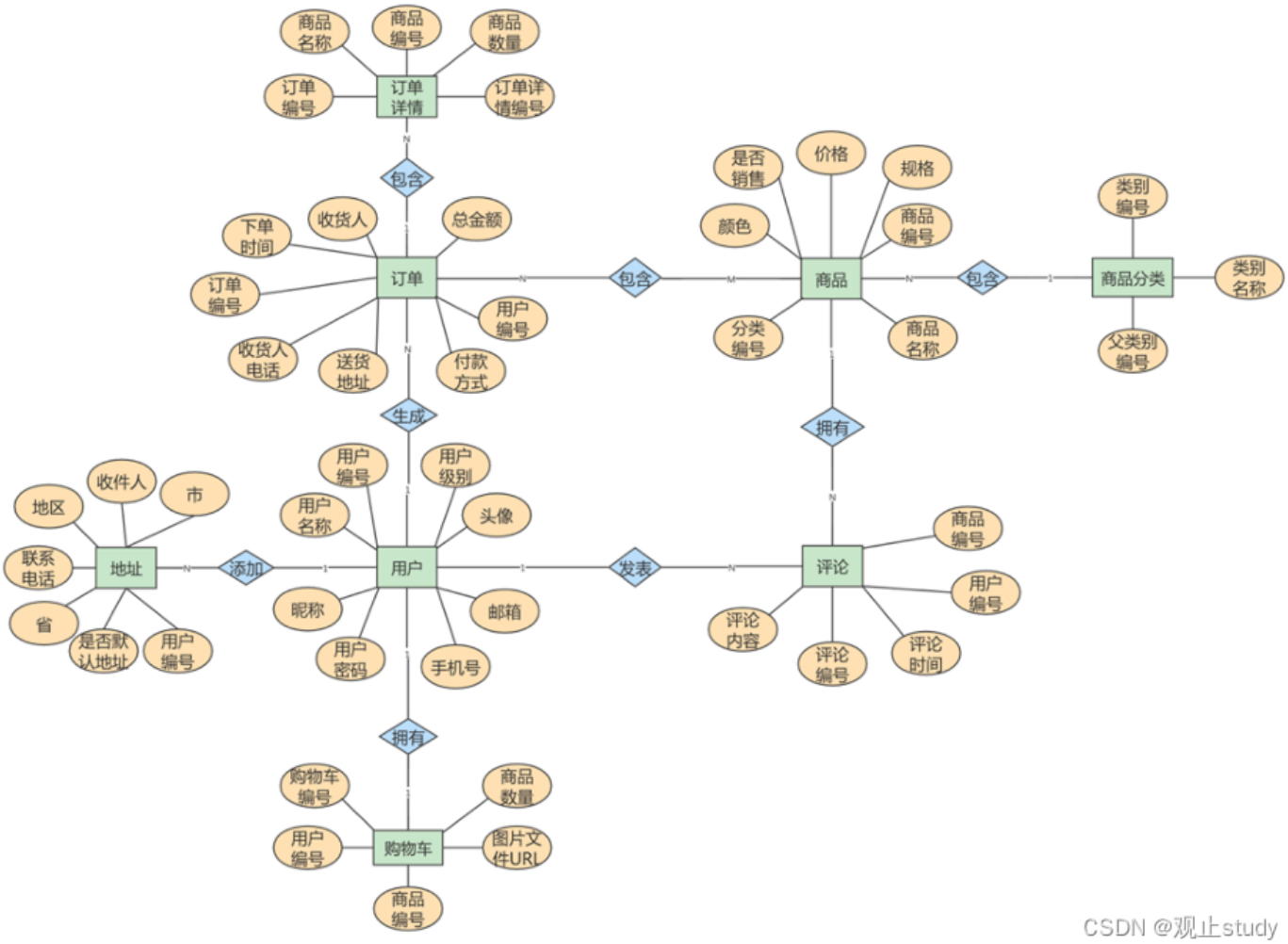
根据实际场景我们可以分析出如下关联关系：



- 用户可以在电商平台添加多个地址；
- 用户只能拥有一个购物车；
- 用户可以生成多个订单；
- 用户可以发表多条评论；
- 一件商品可以有多条评论；
- 每一个商品分类包含多种商品；
- 一个订单可以包含多个商品，一个商品可以在多个订单里。
- 订单中又包含多个订单详情，因为一个订单中可能包含不同种类的商品；

(3) 模型细化

有了上述 ER 模型，我们就可以从整体上理解各个实体之间所在存在的关联关系。为了能更清晰的查看各个实体对应到具体的表的关联关系。我们还需要把各个实体的属性加上。



- **地址**实体：用户编号、省、市、地区、收件人、联系电话、是否是默认地址。
- **用户**实体：用户编号、用户名称、昵称、用户密码、手机号、邮箱、头像、用户级别。
- **购物车**实体：购物车编号、用户编号、商品编号、商品数量、图片文件url。
- **订单**实体：订单编号、收货人、收件人电话、总金额、用户编号、付款方式、送货地址、下单时间。
- **订单详情**实体：订单详情编号、订单编号、商品名称、商品编号、商品数量。
- **商品**实体：商品编号、价格、商品名称、分类编号、是否销售，规格、颜色。
- **评论**实体：评论id、评论内容、评论时间、用户编号、商品编号
- **商品分类**实体：类别编号、类别名称、父类别编号

(4) ER 模型图转换成数据表

通过绘制ER模型，我们已经理清了整体业务逻辑，检查无误后，我们就可以把绘制好的ER模型转换成具体的数据表，转换原则:

- 一个实体通常转换成一个数据表;
- 一个多对多的关系，通常也转换成一个数据表;
- 一个1对1，或者1对多的关系，往往通过表的外键来表达，而不是设计一个新的数据表;
- 属性转换成表的字段。

1. 我们应该先将强实体转换为数据表，因为强实体不依赖于其他的实体而存在。

创建用户信息表：

```
CREATE TABLE user_info(
id bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
user_name varchar (200) DEFAULT NULL COMMENT '用户名称',
nick_name  varchar(280) DEFAULT NULL COMMENT '用户昵称',
passwd varchar (200) DEFAULT NULL COMMENT '用户密码',
phone_num varchar(280) DEFAULT NULL COMMENT '手机号',
email varchar(200) DEFAULT NULL COMMENT '邮箱',
head_img varchar(200) DEFAULT NULL COMMENT '头像',
user_level varchar(200) DEFAULT NULL COMMENT '用户级别',
PRIMARY KEY ( `id ` )
)ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COMMENT='用户表';
```

商品分类实体转换成商品分类表(base_category)，由于商品分类可以有一级分类和二级分类，比如一级分类有家居、手机等等分类，二级分类可以根据手机的一级分类分为手机配件，运营商等，这里我们把商品分类实体规划为两张表，分别是一级分类表和二级分类表，之所以这么规划是因为一级分类和二级分类都是有限的，存储为两张表业务结构更加清晰。

```
-- 一级分类表
CREATE TABLE base_category1(
id bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
name varchar (10) NOT NULL COMMENT '一级分类名称',
PRIMARY KEY (id) USING BTREE
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 ROW_FORMAT=DYNANIC COMMENT='一级分类表';

-- 二级分类表
CREATE TABLE base_category2(
id bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
name varchar (280) NOT NULL COMMENT '二级分类名称',
category1_id bigint(20)DEFAULT NULL COMMENT '一级分类编号',
PRIMARY KEY (id) USING BTREE
)ENGINE=InnODB AUTO_INCRENENT=1 DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='二级分类表';
```

那么如果规划为一张表呢，表结构如下所示。

```
CREATE TABLE base_category(
id bigint(20)NOT NULL AUTO_INCREMENT COMMENT '编号',
name varchar(200) NOT NULL COMMENT '分类名称',
category_parent_id  bigint(20) DEFAULT NULL COMMENT '父分类编号',
PRIMARY KEY (id) USING BTREE
)ENGINE=InnoDB AUTO_INCRENENT=1 DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT= '分类表';
```

如果这样分类的话，那么查询一级分类时候，就需要判断父分类编号是否为空，但是如果插入二级分类的时候也是空，就容易造成业务数据混乱。而且查询二级分类的时候IS NOT NULL条件是无法使用到索引的。同时，这样的设计也不符合第二范式(因为父分类编号并不依赖分类编号ID，因为父分类编号可以有很多数据为NULL)，所以需要进行表的拆分。

2. 接下来就是把剩下的弱实体转换为数据库表,只举例其中一个

```
-- 地址实体转换成地址表(user_address)，如下所示。
CREATE TABLE user_address(
id bigint(20)NOT NULL AUTO_INCREMENT COMMENT '编号',
province varchar (500) DEFAULT NULL COMMENT '省',
city varchar (500) DEFAULT NULL COMMENT '市',
user_address varchar(500) DEFAULT NULL COMMENT'具体地址',
user_id bigint(20) DEFAULT NULL COMMENT '用户id ',
consignee varchar (40) DEFAULT NULL COMMENT '收件人',
phone_num varchar(40) DEFAULT NULL COMMENT'联系方式',
is_default varchar(1) DEFAULT NULL COMMENT'是否是默认',
PRIMARY KEY (id)
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='用户地址表';
```

3. 一个多对多的关系转换成一个数据库表

这个ER模型中的多对多的关系有1个，即商品和订单之间的关系，同品类的商品可以出现在不同的订单中，不同的订单也可以包含同一类型的商品，所以它们之间的关系是多对多。针对这种情况需要设计一个独立的表来表示，这种表一般称为中间表。
我们可以设计一个独立的订单详情表，来代表商品和订单之间的包含关系。这个表关联到2个实体，分别是订单、商品。所以，表中必须要包括这两个实体转换成的表的主键。除此之外，我们还要包括该关系自有的属性:商品数量，商品下单价格以及商品名称。

```
-- 订单详情表
CREATE TABLE order_detail(
id  bigint(20) NOT NULL AUTO_INCREMENT COMMENT '订单详情编号',
order_id  bigint(20) DEFAULT NULL COMMENT '订单编号',
sku_id  bigint(20) DEFAULT NULL COMMENT 'sku_id',
sku_name  varchar(20) DEFAULT NULL COMMENT 'sku名称',
sku_num  varchar(200) DEFAULT NULL COMMENT '购买个数',
create_time  datetime DEFAULT NULL COMMENT '操作时间',
PRIMARY KEY (id) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='订单明细表';
```

4. 通过外键来描述一对多关系

在上面的表的设计中，我们可以用外键来表达1对多的关系。比如在商品评论表sku_comments中，我们分别把user_id、sku_id定义成外键，以使用下面的语句设置外键。

```
CONSTRAINT fk_comment_user FOREIGN KEY (user_id) REFERENCES user_info (id),
CONSTRAINT fk_comment.sku FOREIGN KEY (sku_id) REFERENCES sku_info ( sku_id)
```

外键约束主要是在数据库层面上保证数据的一致性，但是因为插入和更新数据需要检查外键，理论上性能会有所下降。实际的项目，不建议使用外键，一方面是提高了开发的复杂度(有外键的话主从表类的操作必须先操作主表)，此外在有外键时,处理数据的时候非常麻烦。我们通常会选择在应用层面做数据的一致性检查。如学生选课的场景，课程不应该是有学生自己输入，而是通过下拉或查找等方式从系统中进行选取，就能够保证是合法的课程ID，因此就不需要靠数据库的外键来检查了。

五.全文概览