1. 数据库设计三大范式

为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。

1. 第一范式(确保每列保持原子性)

第一范式是最基本的范式。如果数据库表中的所有字段值都是不可分解的原子值，就满足第一范式。

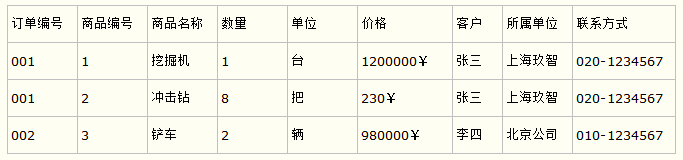
**第一范式的合理遵循需要根据系统的实际需求来定。**比如某些数据库系统中需要用到“地址”这个属性，本来直接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操作的时候将非常方便。这样设计才算满足了数据库的第一范式，如下表所示。



1. 第二范式（主键列与非主键列遵循完全函数依赖关系）

第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中只能保存一种数据，不可以把多种数据保存在同一张数据库表。每一行的数据只能与其中一列相关，即一行数据只做一件事。只要数据列中出现数据重复，就要把表拆分开来。

1. 比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联合主键，如下表所示。



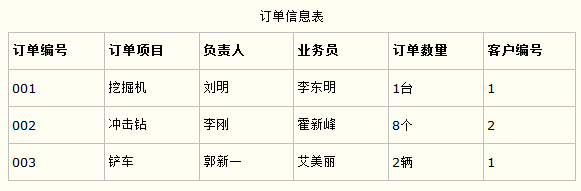
这样就产生一个问题：这个表中是以订单编号和商品编号作为联合主键。这样在该表中商品名称、单位、商品价格等信息不与该表的主键相关，而仅仅是与商品编号相关。所以在这里违反了第二范式的设计原则。

而如果把这个订单信息表进行拆分，把商品信息分离到另一个表中，把订单项目表也分离到另一个表中，就非常完美了。如下所示。



1. 第三范式（非主键列之间没有传递函数依赖关系）

非主键列之间不会相互影响到，降低耦合度。





二、E-R图

1. 概念

E-R图也称实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。

2. ER图分为实体、属性、关系三个核心部分。实体是长方形体现，而属性则是椭圆形，关系为菱形。

(1) ER图的实体（entity）即数据模型中的数据对象，例如人、学生、音乐都可以作为一个数据对象，用长方体来表示，每个实体都有自己的实体成员（entity member）或者说实体对象（entity instance），例如学生实体里包括张三、李四等，实体成员（entity member）/实体实例（entity instance） 不需要出现在ER图中。

(2) ER图的属性（attribute）即数据对象所具有的属性，例如学生具有姓名、学号、年级等属性，用椭圆形表示，属性分为唯一属性（ unique attribute）和非唯一属性，唯一属性指的是唯一可用来标识该实体实例或者成员的属性，用下划线表示，一般来讲实体都至少有一个唯一属性。

(3) ER图的关系（relationship）用来表现数据对象与数据对象之间的联系，例如学生的实体和成绩表的实体之间有一定的联系，每个学生都有自己的成绩表，这就是一种关系，关系用菱形来表示。

3. ER图中关联关系有三种：

(1) 1对1（1:1） ：1对1关系是指对于实体集A与实体集B，A中的每一个实体至多与B中一个实体有关系；反之，在实体集B中的每个实体至多与实体集A中一个实体有关系。

(2) 1对多（1:N） ：1对多关系是指实体集A与实体集B中至少有N(N>0)个实体有关系；并且实体集B中每一个实体至多与实体集A中一个实体有关系。

(3) 多对多（M:N） ：多对多关系是指实体集A中的每一个实体与实体集B中至少有M(M>0)个实体有关系，并且实体集B中的每一个实体与实体集A中的至少N（N>0）个实体有关系。

4. 例子

