一些术语：
relation —— 关系模型
tuple entity—— 表中的一条记录，也成为实体
attribute/column —— 每个表头即属性
domain —— 属性的取值域
FD（functional dependency）—— 函数依赖
MVD(Multi-Valued Dependency) —— 多值依赖

Concepts of ER Model

- Entities　the real world object
- Attributes like customer name
- Relationship

Type of Key

- Primary key
- Candidate key
- Foreign key

Normalization of Database

数据库正规化是数据库中如何合理且有效组织数据的一门技术，它也为我们提供了一套系统地消除数据冗余（redundancy）和异常（anomaly）的方法。

正规化主要就是两个目的：

消除数据冗余（或者说没意义的数据）
确保数据之间的依赖关系是合乎逻辑的
Problems Without Normalization

假如在设计数据库关系的时候没考虑正规化，不仅会因为数据间的冗余从而浪费存储空间，同时，它也会给我们更新、删除数据时带来很多麻烦，也就是上面说到的异常，下面举例说明。

表头分别是学生id，学生姓名，学生具体的专业方向，hod(Head of Department) 学院的领导，办公电话。

插入异常

假如来了个新生，他还没细分具体专业方向，于是乎在后三栏我们就没办法添了，只能用NULL来补充了，这样就很不方便了；相反，假如软件工程专业有50个学生，那么这50个学生的后三栏就会重复，但我们不得不重复地插入。

更新异常

好，假如某天Mr. X不当领导了，然后换了个领导Mr. Y，那么每个学生都需要将他的hod改成Mr. Y了。万一数据库管理员不小心漏了几个同学的没改，那这些学生的学籍就有错误了。

也许你会说：哎呀，那你管理员细心一点不就好了？

那我会说：你为什么不可以换一种设计数据表的方式呢？

删除异常

再假设，某个branch只有一个学生（好像不太可能），有一天，这个学生真的太孤单了，他读不下去了，转去学管理了，很显然就要把他的信息删掉，问题就来了。因为我们的学生信息和专业的信息是绑在一起的，你删了这个唯一的同学，那这个专业、专业领导也被你删掉了，这样就有问题了。

First Normal Form (1NF) https://blog.csdn.net/qq_37174526/article/details/82776507

如果一个表是满足一范式的，那么它需要满足以下条件：

- 每个属性的取值必须是原子的atomic
- 每一列的domain必须是一样的
- 不存在相同的列名
- 列的排列顺序，以及每个tuple间的排列顺序可以交换。

Second Normal Form (2NF) https://blog.csdn.net/qq_37174526/article/details/84107366

如果一个表是满足二范式的，那么它需要满足以下条件：

- 它必须是满足一范式的
- 它不含有部分依赖（Partial Dependency）

Third Normal Form (3NF) https://blog.csdn.net/qq_37174526/article/details/84107958

如果一个表是满足三范式的，那么它需要满足以下条件：

- 它必须是满足二范式的
- 它不含有传递依赖（TransitiveDependency）

区别identifying和non-identifying

图书表的“Auth_id”作为外键码约束条件引用作者表的主键码，说得通俗点就是说作者表（Authors）是父表，图书表（Booklist）是子表。要想知道图书表中《WordPress技巧》的作者是谁，需要通过字段“Auth_id”在作者表中查询，也就是说Auth_id是图书表的外键码。

> 即父记录在子记录创建之前创建，子记录在父记录删除之前删除。

例如要在图书表添加《CSS实战》这本图书，首先要在作者表先添加作者“yangjf”；而如果要在作者表中删除作者“yangjf”这本书，首先要删除图书表中《CSS实战》这本书，从而保证数据的引用完整性，也就避免出现无父记录的“孤儿”记录。

说了那么多，那么以上两个表究竟是“Identifying relationship”关系，还是“non-identifying relationship”关系呢？答案取决于这个图书管理系统的业务规则。

1、当图书管理系统的业务规则规定：作者必须有图书，没有图书的作者不能录入系统中。

那么作者表（Authors）与图书表（Booklist）的关联关系为“Identifying relationship”。

也就是说“identifying relationship”表示如果父表有记录，那么子表中必须有记录。即作者表（Authors）中的每一位作者在图书表（Booklist）中至少存在一本书（当然可以有多本书，因为一个作者完全有可能有多本书）。

同理，“identifying relationship”关联关系还表示如果父表没有记录，则子表也不能有记录。例如如果作者表中没有作者“yangjf”的信息，那么图书表中就不能有图书《CSS实战》。

2、当图书管理系统的业务规则规定：作者可以没有图书。

那么作者表（Authors）与图书表（Booklist）的关联关系为“non-identifying relationship”。

也就是说“non-identifying relationship”表示父表的数据可以独立于子表的数据存在

一句话总结：父表的主键在子表中作主键就是identifying，不作主键就是non-identifying