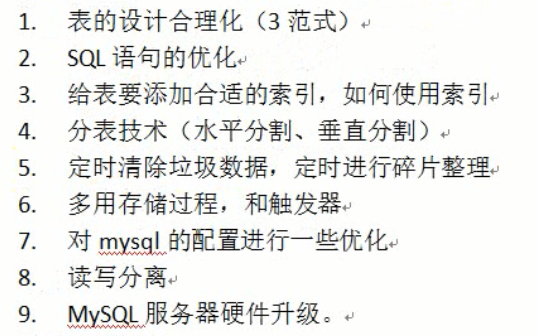
数据库优化是一个综合型的技术，并不是通过某一种方式让数据库效率提高很多，而是通过多方面的提高，从而使得数据库提高很多。

## 数据库优化包括的方面



## 数据库的设计

#### 为什么要设计数据库

良好的数据库：

1. 节省存储空间
2. 保证数据完整性

糟糕的数据库：

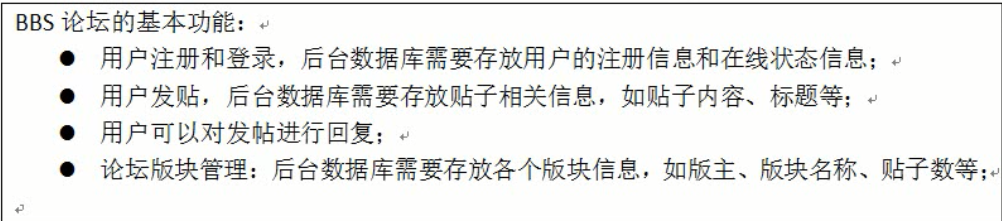
1. 数据冗余，存储空间的浪费
2. 产生数据不完整

#### 设计数据的步骤

1. 充分了解需求
2. 标识实体：具体存在的对象，名词
3. 标识属性
4. 标识关系

例题：

（1）需求如下：



（2）实体：用户、发帖、回帖、版块

一个实体就是一张表，每个对象都有属性，每个属性都是一个字段

（3）属性如下：



（4）关系：

表与表之间的关系：一对一、一对多、多对一、多对多

如何实现：

一对一：两个公共字段都是主键，则为一对一关系，因为主键不允许重复



一对多：主表中的一条记录对应从表中的多条记录，则为一对多（主键和非主键的关系）

多对一：非主键和主键的关系

多对多：非主键和非主键的关系

红体为常用关系

#### 如何将E-R图转换为表

1. 将实体转成表，将属性转成字段
2. 如果找不到一个合适的字段做主键，我们就添加一个自动增长列作为主键，如上例中的设计：



1. 建立正确的关系。

#### 数据规范化

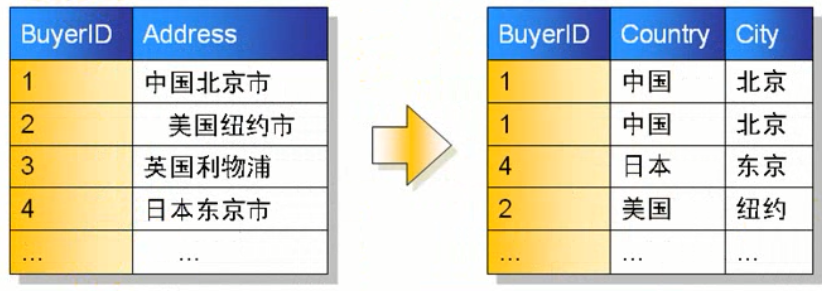
仅仅设计得好，还不足以避免数据冗余，接下来还需要做数据的规范化。

表设计出来，并不是最合理的结构，我们需要对表进行规范化，通过三范式来对表进行规范化。（本有6范式，范式越多冗余越少，但查询效率越低，总体呈一个正态分布图，当三范式时最优）

何为三范式：（1NF、2NF、3NF）先满足第一范式，再以此类推，逐层追加。

1. 确保每列的原子性（用来规范化所有的字段，所有的字段都不可再分。全约束）

如下图所示：



在实际应用中如果生搬硬套三范式，可能会造成效率的低下，一切从需求出发。



邮寄地址是否要拆分需要看实际业务中是否需要按维度进行分析，地址这个字段，如果不分类汇总，不排序，仅仅是起一个字符串的作用，这时不拆分，这是反三范式。

1. 非键字段必须依赖与键字段

键字段为主键字段，一个表只描述一件事，只描述一个对象。



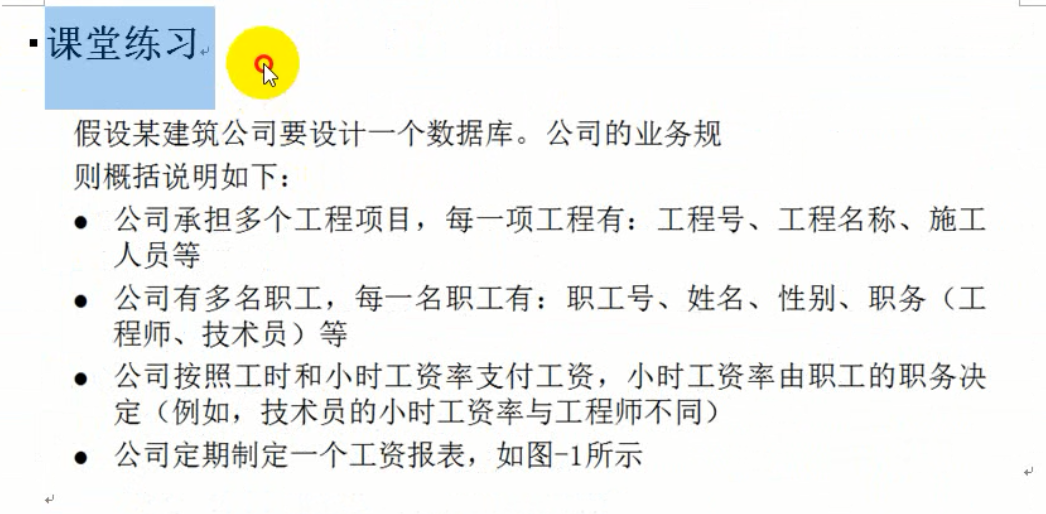
1. 消除传递依赖

只针对非主键部分的约束，在非主键字段中，如果一个字段可以推导出另一个字段，则为传递依赖。例如：



学号和姓名产生了数据冗余，因为通过学号（主键）可以推导出姓名是什么，这就是传递依赖；如果学号不是主键可以推导出姓名是什么，就不是传递依赖。

规范化实例：



公司的工资报表如下：



工资报表做处理：



思考：能否把这个表做数据表用来存储工资信息？（如果增删改没问题就可以，如果有问题就不能用）

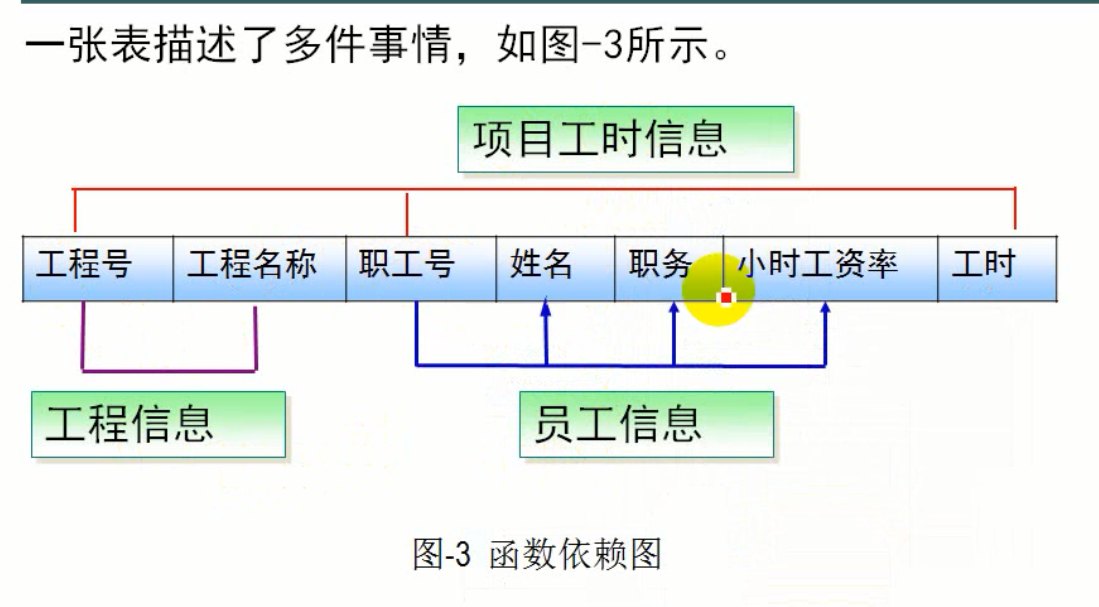
增：如果今天张三入职了，那么他不可能有工程号和工程名称，然而工程号和工程名称必须存在，如何判断该字段能否为空（姓名不能为空、性别不能为空、地址不能为空，地址不详，成绩可以为空）。

改：如李思岐升职为工程师，但其工资未变，而错误。

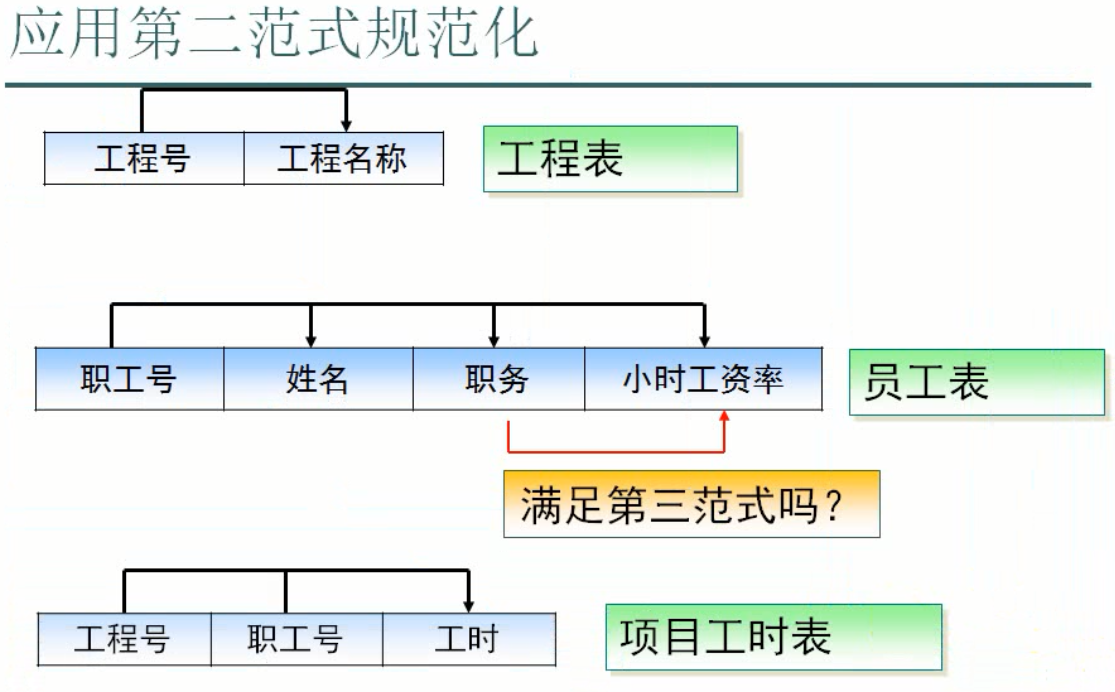
删：如李光明和李思岐不干了，删掉，则花园大厦工程都没了，而错误。

从三范式角度分析过程如下：

第一范式：



应用第二范式规范化：



应用第三范式规范化：

