**数据库三范式经典实例解析**database

[数据结构](http://www.iteye.com/blogs/tag/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84)[编程](http://www.iteye.com/blogs/tag/%E7%BC%96%E7%A8%8B)[制造](http://www.iteye.com/blogs/tag/%E5%88%B6%E9%80%A0)

**数据库三范式经典实例解析**

**引言**

数据库的设计范式是数据库设计所需要满足的规范，满足这些规范的数据库是简洁的、结构明晰的，同时，不会发生插入（insert）、删除（delete）和更新（update）操作异常。反之则是乱七八糟，不仅给数据库的编程人员制造麻烦，而且面目可憎，可能存储了大量不需要的冗余信息。  
     设计范式是不是很难懂呢？非也，大学教材上给我们一堆数学公式我们当然看不懂，也记不住。所以我们很多人就根本不按照范式来设计数据库。  
     实质上，设计范式用很形象、很简洁的话语就能说清楚，道明白。本文将对范式进行通俗地说明，并以笔者曾经设计的一个简单论坛的数据库为例来讲解怎样将这些范式应用于实际工程。

**范式说明  
     第一范式（1NF）：数据库表中的字段都是单一属性的，不可再分。这个单一属性由基本类型构成，包括整型、实数、字符型、逻辑型、日期型等。  
     例如，如下的数据库表是符合第一范式：**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段1 | 字段2 | 字段3 | 字段4 |
| ? | ? | ? | ? |

**而这样的数据库表是不符合第一范式的：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段1 | 字段2 | 字段3 | | 字段4 |
| ? | ? | 字段3.1 | 字段3.2 | ? |

**很显然，在当前的任何关系数据库管理系统（DBMS）中，傻瓜也不可能做出不符合第一范式的数据库，因为这些DBMS不允许你把数据库表的一列再分成二列或多列。因此，你想在现有的DBMS中设计出不符合第一范式的数据库都是不可能的。  
     第二范式（2NF）：数据库表中不存在非关键字段对任一候选关键字段的部分函数依赖（部分函数依赖指的是存在组合关键字中的某些字段决定非关键字段的情况），也即所有非关键字段都完全依赖于任意一组候选关键字。  
     假定选课关系表为SelectCourse(学号, 姓名, 年龄, 课程名称, 成绩, 学分)，关键字为组合关键字(学号, 课程名称)，因为存在如下决定关系：  
     (学号, 课程名称) → (姓名, 年龄, 成绩, 学分)  
     这个数据库表不满足第二范式，因为存在如下决定关系：  
     (课程名称) → (学分)  
     (学号) → (姓名, 年龄)  
即存在组合关键字中的字段决定非关键字的情况。  
     由于不符合2NF，这个选课关系表会存在如下问题：  
     (1) 数据冗余：  
     同一门课程由n个学生选修，"学分"就重复n-1次；同一个学生选修了m门课程，姓名和年龄就重复了m-1次。  
     (2) 更新异常：  
     若调整了某门课程的学分，数据表中所有行的"学分"值都要更新，否则会出现同一门课程学分不同的情况。  
     (3) 插入异常：  
     假设要开设一门新的课程，暂时还没有人选修。这样，由于还没有"学号"关键字，课程名称和学分也无法记录入数据库。  
     (4) 删除异常：  
     假设一批学生已经完成课程的选修，这些选修记录就应该从数据库表中删除。但是，与此同时，课程名称和学分信息也被删除了。很显然，这也会导致插入异常。**

**把选课关系表SelectCourse改为如下三个表：  
     学生：Student(学号, 姓名, 年龄)；  
     课程：Course(课程名称, 学分)；  
     选课关系：SelectCourse(学号, 课程名称, 成绩)。  
     这样的数据库表是符合第二范式的，消除了数据冗余、更新异常、插入异常和删除异常。  
     另外，所有单关键字的数据库表都符合第二范式，因为不可能存在组合关键字。  
     第三范式（3NF）：在第二范式的基础上，数据表中如果不存在非关键字段对任一候选关键字段的传递函数依赖则符合第三范式。所谓传递函数依赖，指的是如果存在"A → B → C"的决定关系，则C传递函数依赖于A。因此，满足第三范式的数据库表应该不存在如下依赖关系：  
     关键字段 → 非关键字段x → 非关键字段y  
     假定学生关系表为Student(学号, 姓名, 年龄, 所在学院, 学院地点, 学院电话)，关键字为单一关键字"学号"，因为存在如下决定关系：  
     (学号) → (姓名, 年龄, 所在学院, 学院地点, 学院电话)  
这个数据库是符合2NF的，但是不符合3NF，因为存在如下决定关系：  
     (学号) → (所在学院) → (学院地点, 学院电话)  
即存在非关键字段"学院地点"、"学院电话"对关键字段"学号"的传递函数依赖。  
     它也会存在数据冗余、更新异常、插入异常和删除异常的情况，读者可自行分析得知。  
     把学生关系表分为如下两个表：  
     学生：(学号, 姓名, 年龄, 所在学院)；  
     学院：(学院, 地点, 电话)。  
这样的数据库表是符合第三范式的，消除了数据冗余、更新异常、插入异常和删除异常。  
     鲍依斯-科得范式（BCNF）：在第三范式的基础上，数据库表中如果不存在任何字段对任一候选关键字段的传递函数依赖则符合第三范式。  
     假设仓库管理关系表为StorehouseManage(仓库ID, 存储物品ID, 管理员ID, 数量)，且有一个管理员只在一个仓库工作；一个仓库可以存储多种物品。这个数据库表中存在如下决定关系：  
     (仓库ID, 存储物品ID) →(管理员ID, 数量)  
     (管理员ID, 存储物品ID) → (仓库ID, 数量)  
     所以，(仓库ID, 存储物品ID)和(管理员ID, 存储物品ID)都是StorehouseManage的候选关键字，表中的唯一非关键字段为数量，它是符合第三范式的。但是，由于存在如下决定关系：  
     (仓库ID) → (管理员ID)  
     (管理员ID) → (仓库ID)  
即存在关键字段决定关键字段的情况，所以其不符合BCNF范式。它会出现如下异常情况：  
     (1) 删除异常：  
     当仓库被清空后，所有"存储物品ID"和"数量"信息被删除的同时，"仓库ID"和"管理员ID"信息也被删除了。  
     (2) 插入异常：  
     当仓库没有存储任何物品时，无法给仓库分配管理员。  
     (3) 更新异常：  
     如果仓库换了管理员，则表中所有行的管理员ID都要修改。  
     把仓库管理关系表分解为二个关系表：  
     仓库管理：StorehouseManage(仓库ID, 管理员ID)；  
     仓库：Storehouse(仓库ID, 存储物品ID, 数量)。  
     这样的数据库表是符合BCNF范式的，消除了删除异常、插入异常和更新异常。**

**范式应用  
     我们来逐步搞定一个论坛的数据库，有如下信息：  
     （1） 用户：用户名，email，主页，电话，联系地址  
     （2） 帖子：发帖标题，发帖内容，回复标题，回复内容   
     第一次我们将数据库设计为仅仅存在表：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 用户名 | email | 主页 | 电话 | 联系地址 | 发帖标题 | 发帖内容 | 回复标题 | 回复内容 |

**这个数据库表符合第一范式，但是没有任何一组候选关键字能决定数据库表的整行，唯一的关键字段用户名也不能完全决定整个元组。我们需要增加"发帖ID"、"回复ID"字段，即将表修改为：**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 用户名 | email | 主页 | 电话 | 联系地址 | 发帖ID | 发帖标题 | 发帖内容 | 回复ID | 回复标题 | 回复内容 |

**这样数据表中的关键字(用户名，发帖ID，回复ID)能决定整行：  
     (用户名,发帖ID,回复ID) → (email,主页,电话,联系地址,发帖标题,发帖内容,回复标题,回复内容)  
但是，这样的设计不符合第二范式，因为存在如下决定关系：  
     (用户名) → (email,主页,电话,联系地址)  
     (发帖ID) → (发帖标题,发帖内容)  
     (回复ID) → (回复标题,回复内容)  
即非关键字段部分函数依赖于候选关键字段，很明显，这个设计会导致大量的数据冗余和操作异常。  
     我们将数据库表分解为（带下划线的为关键字）：  
     （1） 用户信息：用户名，email，主页，电话，联系地址  
     （2） 帖子信息：发帖ID，标题，内容  
     （3） 回复信息：回复ID，标题，内容**

**（4） 发贴：用户名，发帖ID  
     （5） 回复：发帖ID，回复ID  
     这样的设计是满足第1、2、3范式和BCNF范式要求的，但是这样的设计是不是最好的呢？  
     不一定。  
     观察可知，第4项"发帖"中的"用户名"和"发帖ID"之间是1：N的关系，因此我们可以把"发帖"合并到第2项的"帖子信息"中；第5项"回复"中的"发帖ID"和"回复ID"之间也是1：N的关系，因此我们可以把"回复"合并到第3项的"回复信息"中。这样可以一定量地减少数据冗余，新的设计为：  
     （1） 用户信息：用户名，email，主页，电话，联系地址  
     （2） 帖子信息：用户名，发帖ID，标题，内容  
     （3） 回复信息：发帖ID，回复ID，标题，内容  
     数据库表1显然满足所有范式的要求；  
     数据库表2中存在非关键字段"标题"、"内容"对关键字段"发帖ID"的部分函数依赖，即不满足第二范式的要求，但是这一设计并不会导致数据冗余和操作异常；  
     数据库表3中也存在非关键字段"标题"、"内容"对关键字段"回复ID"的部分函数依赖，也不满足第二范式的要求，但是与数据库表2相似，这一设计也不会导致数据冗余和操作异常。  
     由此可以看出，并不一定要强行满足范式的要求，对于1：N关系，当1的一边合并到N的那边后，N的那边就不再满足第二范式了，但是这种设计反而比较好！  
     对于M：N的关系，不能将M一边或N一边合并到另一边去，这样会导致不符合范式要求，同时导致操作异常和数据冗余。   
对于1：1的关系，我们可以将左边的1或者右边的1合并到另一边去，设计导致不符合范式要求，但是并不会导致操作异常和数据冗余。  
  
结论  
     满足范式要求的数据库设计是结构清晰的，同时可避免数据冗余和操作异常。这并意味着不符合范式要求的设计一定是错误的，在数据库表中存在1：1或1：N关系这种较特殊的情况下，合并导致的不符合范式要求反而是合理的。  
     在我们设计数据库的时候，一定要时刻考虑范式的要求。**

**数据库三范式，轻松理解**

来源：互联网 作者：古嗣小井 发表于：2009-09-29 10:59　 点击： 34414

网上搜罗了一大堆关于数据库范式理解的文章，都是千律一篇的复制粘贴，连例子都是一模一样，拜托有点创意好不，实在看不下去，自己写一篇个人理解三范式的文章。如果有理解上的不正确之处，请联系我：279537592＃qq.com (#=@) 官方定义：第一范式（1NF）：数

网上搜罗了一大堆关于数据库范式理解的文章，都是千律一篇的复制粘贴，连例子都是一模一样，拜托有点创意好不，实在看不下去，自己写一篇个人理解三范式的文章。如果有理解上的不正确之处，请联系我：279537592＃qq.com (#=>@)

官方定义：第一范式（1NF）：数据库表中的字段都是单一属性的，不可再分。  
我的理解：第一范式这个不用說了，只要是关系数据库都满足第一范式

官方定义：第二范式（2NF）：数据库表中不存在非关键字段对任一候选关键字段的部分函数依赖  
我的理解：在第二范式中组合主键（AB）【注明：也叫做复合主键】里面的A或者B 与其他字段不能存在组合重复，为解决这个问题，通常的做法是咱们不用组合主键，添加一个ID，做为单一主键即可满足第二范式。如果不想添加ID，请满足组合主键（AB）里面的A或者B 与其他字段不能存在组合重复。  
如:不满足第二范式,复合主键中的A与字段C组合重复  
+------------+-----------+-------------------+  
pk pk row  
+------------+-----------+-------------------+  
A B C  
+------------+-----------+-------------------+  
A D C  
+------------+-----------+-------------------+  
A E C  
+------------+-----------+-------------------+

改为这样满足第二范式(但是不满足第三范式,字段A与字段C是组合重复):  
+---------+------------+-----------+-------------------+  
pk row row row  
+---------+------------+-----------+-------------------+  
1 A B C  
+---------+------------+-----------+-------------------+  
2 A D C  
+---------+------------+-----------+-------------------+  
3 A E C  
+---------+------------+-----------+-------------------+

官方定义：第三范式（3NF）：在第二范式的基础上，数据表中如果不存在非关键字段对任一候选关键字段的传递函数依赖则符合第三 范式。  
我的理解：在第三范式中字段与字段之间不能存在组合重复  
如:不满足第三范式,字段A与字段C组合重复  
+---------+------------+-----------+-------------------+---------------+  
pk row row row row  
+---------+------------+-----------+-------------------+---------------+  
1 A B C F  
+---------+------------+-----------+-------------------+---------------+  
2 A D C G  
+---------+------------+-----------+-------------------+---------------+  
3 A E C K  
+---------+------------+-----------+-------------------+---------------+

改为这样满足第三范式:  
表1  
+---------+------------+-----------+  
pk row row   
+---------+------------+-----------+  
1 A B   
+---------+------------+-----------+  
2 A D   
+---------+------------+-----------+  
3 A E   
+---------+------------+-----------+  
和表2  
+---------+-------------------+------------+  
pk row row  
+---------+-------------------+------------+  
1 C F  
+---------+-------------------+------------+  
2 C G  
+---------+-------------------+------------+  
3 C K  
+---------+-------------------+------------+

原则:当出现字段与字段的组合重复,如上的A和C的组合重复,首先要考虑的就是把他们拆分为2个表,具体是C拆到表1, 还是A拆到表1,看情况而定.

关键要理解定义这种范式标准的主要目的是为了减少数据冗余，数据冗余产生的本质就是在一个表中存在字段与字段之间的一对多，或者多对多关系。解决这个几对几的关系问题，就能轻易实现满足第三范式的数据库设计。

|  |
| --- |
| 对数据库三大范式的理解 |
| By  傅健 发表于 2008-8-27 16:49:00 |
| **2**  [推荐](javascript:void(null))  第一范式（1NF）：字段不能划分成更多字段；  不符合第一范式的例子： 表：字段1  字段2  字段3  字段4 字段3.1 字段3.2  现有的DBMS中设计出不符合第一范式的数据库都是不可能的。  第二范式（2NF）：单关键字的表，或者若为组合关键字则必须没有候选关键字段→非关键字段的表；  不符合第二范式的例子： 表：学号, 姓名, 年龄, 课程名称, 成绩, 学分 (课程名称) → (学分) (学号) → (姓名, 年龄) 存在问题： 数据冗余，每条记录都含有相同信息； 删除异常：删除所有学生成绩，就把课程信息全删除了； 插入异常：学生未选课，无法记录进数据库； 更新异常：调整课程学分，所有行都调整。  修正： 学生：Student(学号, 姓名, 年龄)；  课程：Course(课程名称, 学分)；  选课关系：SelectCourse(学号, 课程名称, 成绩)。  第三范式（3NF）：在第二范式的基础上，数据表中如果不存在传递函数依赖：关键字段 → 非关键字段x → 非关键字段y  不符合第三范式的例子： 学号, 姓名, 年龄, 所在学院, 学院地点, 学院电话，关键字为单一关键字"学号" 这个数据库是符合2NF的，但是不符合3NF，因为存在如下决定关系： (学号) → (所在学院) → (学院地点, 学院电话)  修正： 学生：(学号, 姓名, 年龄, 所在学院)； 学院：(学院, 地点, 电话)。 |

轻松理解数据库三范式

一范式1NF

1、    数据库表的每一行都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。

2、    表的每一行包含一个实例的信息。

2.2.    第二范式2NF

1、    要求数据库表中的每个实例或行必须是唯一的。为实现区分，通常需要为表加一个列，以存储各个实例的唯一标识（即主键）。

2、    实体的属性完全依赖于主关键字。所谓完全依赖指不能存在仅依赖主关键字一部分的属性。如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。总之第二范式就是非主属性非部分依赖于主关键字。

2.3.    第三范式3NF

一个数据库表中不包含已在其他表中已包含的非主关键字信息。例如存在一个部门信息表，其中每个部门有部门编号DEPT\_ID、部门名称、部门简介等信息。那么在员工信息表中列出的部门编号DEPT\_ID后，就不能再有关于部门的其他信息，否则就会造成数据冗余。

第一范式（1NF）：在关系模式R中的每一个具体关系r中，如果每个属性值 都是不可再分的最小数据单位，则称R是第一范式的关系。例：如职工号，姓名，电话号码组成一个表（一个人可能有一个办公室电话 和一个家里电话号码） 规范成为1NF有三种方法：

一是重复存储职工号和姓名。这样，关键字只能是电话号码。

二是职工号为关键字，电话号码分为单位电话和住宅电话两个属性

三是职工号为关键字，但强制每条记录只能有一个电话号码。

以上三个方法，第一种方法最不可取，按实际情况选取后两种情况。

第二范式（2NF）：如果关系模式R（U，F）中的所有非主属性都完全依赖于任意一个候选关键字，则称关系R 是属于第二范式的。

例：选课关系 SCI（SNO，CNO，GRADE，CREDIT）其中SNO为学号， CNO为课程号，GRADEGE 为成绩，CREDIT 为学分。 由以上条件，关键字为组合关键字（SNO，CNO）

在应用中使用以上关系模式有以下问题：

a.数据冗余，假设同一门课由40个学生选修，学分就 重复40次。

b.更新异常，若调整了某课程的学分，相应的元组CREDIT值都要更新，有可能会出现同一门课学分不同。

c.插入异常，如计划开新课，由于没人选修，没有学号关键字，只能等有人选修才能把课程和学分存入。

d.删除异常，若学生已经结业，从当前数据库删除选修记录。某些门课程新生尚未选修，则此门课程及学分记录无法保存。

原因：非关键字属性CREDIT仅函数依赖于CNO，也就是CREDIT部分依赖组合关键字（SNO，CNO）而不是完全依赖。

解决方法：分成两个关系模式 SC1（SNO，CNO，GRADE），C2（CNO，CREDIT）。新关系包括两个关系模式，它们之间通过SC1中的外关键字CNO相联系，需要时再进行自然联接，恢复了原来的关系

第三范式（3NF）：如果关系模式R（U，F）中的所有非主属性对任何候选关键字都不存在传递信赖，则称关系R是属于第三范式的。

例：如S1（SNO，SNAME，DNO，DNAME，LOCATION） 各属性分别代表学号，

姓名，所在系，系名称，系地址。

关键字SNO决定各个属性。由于是单个关键字，没有部分依赖的问题，肯定是2NF。但这关系肯定有大量的冗余，有关学生所在的几个属性DNO，DNAME，LOCATION将重复存储，插入，删除和修改时也将产生类似以上例的情况。

原因：关系中存在传递依赖造成的。即SNO -> DNO。 而DNO -> SNO却不存在，DNO -> LOCATION, 因此关键辽 SNO 对 LOCATION 函数决定是通过传递依赖 SNO -> LOCATION 实现的。也就是说，SNO不直接决定非主属性LOCATION。

解决目地：每个关系模式中不能留有传递依赖。

解决方法：分为两个关系 S（SNO，SNAME，DNO），D（DNO，DNAME，LOCATION）

注意：关系S中不能没有外关键字DNO。否则两个关系之间失去联系。

BCNF：如果关系模式R（U，F）的所有属性（包括主属性和非主属性）都不传递依赖于R的任何候选关键字，那么称关系R是属于BCNF的。或是关系模式R，如果每个决定因素都包含关键字（而不是被关键字所包含），则RCNF的关系模式。

例：配件管理关系模式 WPE（WNO，PNO，ENO，QNT）分别表仓库号，配件号，职工号，数量。有以下条件

a.一个仓库有多个职工。

b.一个职工仅在一个仓库工作。

c.每个仓库里一种型号的配件由专人负责，但一个人可以管理几种配件。

d.同一种型号的配件可以分放在几个仓库中。

分析：由以上得 PNO 不能确定QNT，由组合属性（WNO，PNO）来决定，存在函数依赖（WNO，PNO） -> ENO。由于每个仓库里的一种配件由专人负责，而一个人可以管理几种配件，所以有组合属性（WNO，PNO）才能确定负责人，有（WNO，PNO）-> ENO。因为 一个职工仅在一个仓库工作，有ENO -> WNO。由于每个仓库里的一种配件由专人负责，而一个职工仅在一个仓库工作，有 （ENO，PNO）-> QNT。

找一下候选关键字，因为（WNO，PNO） -> QNT，（WNO，PNO）-> ENO ，因此 （WNO，PNO）可以决定整个元组，是一个候选关键字。根据ENO->WNO，（ENO，PNO）->QNT，故（ENO，PNO）也能决定整个元组，为另一个候选关键字。属性ENO，WNO，PNO 均为主属性，只有一个非主属性QNT。它对任何一个候选关键字都是完全函数依赖的，并且是直接依赖，所以该关系模式是3NF。

分析一下主属性。因为ENO->WNO，主属性ENO是WNO的决定因素，但是它本身不是关键字，只是组合关键字的一部分。这就造成主属性WNO对另外一个候选关键字（ENO，PNO）的部 分依赖，因为（ENO，PNO）-> ENO但反过来不成立，而P->WNO，故（ENO，PNO）-> WNO 也是传递依赖。

虽然没有非主属性对候选关键辽的传递依赖，但存在主属性对候选关键字的传递依赖，同样也会带来麻烦。如一个新职工分配到仓库工作，但暂时处于实习阶段，没有独立负责对某些配件的管理任务。由于缺少关键字的一部分PNO而无法插入到该关系中去。又如某个人改成不管配件了去负责安全，则在删除配件的同时该职工也会被删除。

解决办法：分成管理EP（ENO，PNO，QNT），关键字是（ENO，PNO）工作EW（ENO，WNO）其关键字是ENO

缺点：分解后函数依赖的保持性较差。如此例中，由于分解,函数依赖（WNO，PNO）-> ENO 丢失了, 因而对原来的语义有所破坏。没有体现出每个仓库里一种部件由专人负责。有可能出现 一部件由两个人或两个以上的人来同时管理。因此，分解之后的关系模式降低了部分完整性约束。

一个关系分解成多个关系，要使得分解有意义，起码的要求是分解后不丢失原来的信息。这些信息不仅包括数据本身，而且包括由函数依赖所表示的数据之间的相互制约。进行分解的目标是达到更高一级的规范化程度，但是分解的同时必须考虑两个问题：无损联接性和保持函数依赖。有时往往不可能做到既有无损联接性，又完全保持函数依赖。需要根据需要进行权衡。

1NF直到BCNF的四种范式之间有如下关系：

BCNF包含了3NF包含2NF包含1NF

小结：

目地：规范化目的是使结构更合理，消除存储异常，使数据冗余尽量小，便于插入、删除和更新

原则：遵从概念单一化 "一事一地"原则，即一个关系模式描述一个实体或实体间的一种联系。规范的实质就是概念的单一化。

方法：将关系模式投影分解成两个或两个以上的关系模式。

要求：分解后的关系模式集合应当与原关系模式"等价"，即经过自然联接可以恢复原关系而不丢失信息，并保持属性间合理的联系。

注意：一个关系模式结这分解可以得到不同关系模式集合，也就是说分解方法不是唯一的。最小冗余的要求必须以分解后的数据库能够表达原来数据库所有信息为前提来实现。其根本目标是节省存储空间，避免数据不一致性，提高对关系的操作效率，同时满足应用需求。实际上，并不一定要求全部模式都达到BCNF不可。有时故意保留部分冗余可能更方便数据查询。尤其对于那些更新频度不高，查询频度极高的数据库系统更是如此。

在关系数据库中，除了函数依赖之外还有多值依赖，联接依赖的问题，从而提出了第四范式，第五范式等更高一级的规范化要求。在此，以后再谈。

各位朋友，你看过后有何感想，其实，任何一本数据库基础理论的书都会讲这些东西，考虑到很多网友是半途出家，来做数据库。特找一本书大抄特抄一把，各位有什么问题，也别问我了，自已去找一本关系数据库理论的书去看吧，说不定，对各位大有帮助。说是说以上是基础理论的东西，请大家想想，你在做数据库设计的时候有没有考虑过遵过以上几个范式呢，有没有在数据库设计做得不好之时，想一想，对比以上所讲，到底是违反了第几个范式呢？

我见过的数据库设计，很少有人做到很符合以上几个范式的，一般说来，第一范式大家都可以遵守，完全遵守第二第三范式的人很少了，遵守的人一定就是设计数据库的高手了，BCNF的范式出现机会较少，而且会破坏完整性，你可以在做设计之时不考虑它，当然在ORACLE中可通过触发器解决其缺点。以后我们共同做设计之时，也希望大家遵守以上几个范式。

那些数据库的书介绍的数据库范式，实在是晦涩难懂，我在这里给出一个通俗的描述：

1NF：一个table中的列是不可再分的（即列的原子性）

2NF：一个table中的行是可以唯一标示的，（即table中的行是不可以有重复的）

3NF：一个table中列不依赖以另一个table中的非主键的列，还是不通俗！巨寒！！

        举个例子吧：有一个部门的table，我们叫它tbl\_department, 它有这么几列（dept\_id(pk),dept\_name,dept\_memo...） 有一个员工table，我们叫它tbl\_employee,在这个table中有一列dept\_id(fk)描述关于部门的信息，若tbl\_employee要满足3NF，则在tbl\_employee中就不得再有除dept\_id列的其它有关部门信息的列！

一般数据库的设计满足3NF即可！（个人觉得应该尽可能的满足3NF，一家之言^\_^）

BCNF：通常认为BCNF是修正的第三范式，它比3NF又进一步！

4NF：

5NF：将一个table尽可能的分割成小的块，以排除在table中所有冗余的数据

==============================

解释1道数据库规范化/范式的例题?

比如解释例题:有属性SNO(学生号),SNAM(学生姓名),DON(系) MAN(系主任),舍关系R的主键为SNO,则R为1NF\2NF\3NF?

答案是2NF

为什么?

学号是主属性,姓名可以直接被学号给确定,学号可以确定学生所在的系,而系主任不是由主属性学号来确定的,是由系名来确定的,所以问题就在系主任上,在这个关系里学号和系主任之间不是直接确定的而是间接确定的,这就叫非主属性对主属性存在传递函数依赖,所以是2范式,1范式是把两个或几个事在一个关系里说,这个就是稍微好点,可以根据主码确定一个非主属性,还是间接的,所以是第2范式.

因为只要知道姓名就可以查到系，查到班主任，所以姓名并不是只和学生号产生关系。

这是不符合第三范式的。你去仔细研读一下第三范式的定义就能够想通了

其实这种现象就被称为冗余，而且属于复式冗余，因为班主任甚至还和系有关系。

因为只要知道姓名就可以查到系，查到班主任，所以姓名并不是只和学生号产生关系。

这是不符合第三范式的。你去仔细研读一下第三范式的定义就能够想通了

其实这种现象就被称为冗余，而且属于复式冗余，因为班主任甚至还和系有关系。

============================

数据库的规范化,1\2\3范式怎么理解?

所谓1NF,2NF,3NF,各举例说明下吧,让我能理解....

1NF:字段具有原子性，不可再分；

比如说籍贯这个字段，里面是“湖北武汉”的话，它就违反了原子性，因为湖北武汉还可以再分的更具体，分为“湖北”和“武汉”

2NF:组合关键字的表，不存在组合关键字中的任意字段决定其它非关键字段的情部（也就是说不能有两个组合键组成一个主键）

3NF:在2NF的基础上，数据表中如果不存在非关键字段对任一候选字段的传递函数依赖则符合第三范式（也就是说违反了数据冗余）

帐号 身份证号 姓名 密码

1001 410101001 李梅 100001

身份证号和姓名共同决定了密码，姓名是依赖于身份证号的，这样就违反了第三范式

典型的网络结构莫过于MVC结构了，即视图层-模型层-控制层，视图层负责展示页面，就是平时我们上网看到的页面；控制层负责逻辑控制，就是控制我们点击一个链接或点击一个按钮之后，页面要执行什么操作、怎么跳转；业务层就是你所要执行的操作，比如说增、删、改、查询数据，这些操作都是由业务层来完成的。一个好的网络结构一般业务层分为业务层、VO层和数据持久层，业务层负责完成逻辑操作，VO层负责存储数据对象，数据持久层负责数据库操作。所以，页面的数据要经过视图层——〉控制层——〉模型层——〉VO层——〉数据持久层——〉数据库，这样一个流程，数据是一层一层的传递的，不应该跨层操作。

PS：我所说的是一个典型的网站结构，实际上任何一层都可以对数据库进行操作，但是不建议这样做，因为这样做会使层次模糊，不便于模糊与操作