1.范式说明

1.1 第一范式（1NF）无重复的列

所谓第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能同时有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。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。在第一范式（1NF）中表的每一行只包含一个实例的信息。简而言之，第一范式就是无重复的列。

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。在当前的任何关系数据库管理系统（DBMS）中，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。

举例 1：

一张学生表 Student(stuNo,stuName,age,age,sex)是不符合第一范式的，因为有重复列 age 属性。去除重复列 age 以后的 Student(stuNo,stuName,age,sex)是符合第一范式的。

1.2 第二范式（2NF）属性完全依赖于主键 [消除部分子函数依赖]

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。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。例如员工信息表中加上了员工编号（emp_id）列，因为每个员工的员工编号是唯一的，因此每个员工可以被唯一区分。这个唯一属性列被称为主关键字或主键、主码。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是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。简而言之，第二范式就是属性完全依赖于主键。

这里说的主关键字可能不只有一个，有些情况下是存在联合主键的，就是主键有多个属性。

举例 2：

以学生选课为例，每个学生都可以选课，并且有这一门课程的成绩，那么如果将这些信息都放在一张表

StuGrade(stuNo,stuName,age,sex,courseNo,courseNa

me,credit,score)。如果不仔细看，我们会以为这张表的主键是 stuNo，但是当我们看到最后一个 score 属性以后，在想想如果没有课程信息，那么哪里有学生成绩信息呢。所以这张表的主键是一个联合主键(stuNo,corseNo)，这个联合属性能够唯一确定 score 属性。那么再看其他信息，比如 stuName 只需要 stuNo 就能够唯一确定，courseName 只需要 courseNo 就能够唯一确定，因此这样就存在了部分依赖，不符合第二范式。如果要让学生课程成绩信息满足第二范式，那么久需要将这张表拆分成多张表，一张学生表 Studnet(stuNo,stuName,age,sex)，一张课程表 Course(courseNo,courseName,credit)，还有最后一张学生课程成绩表 StuGrade(stuNo,courseNo,score)。这样就符合第二范式了。

1.3 第三范式（3NF）属性不依赖于其它非主属性 [消除传递依赖]

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

举例 3:

每一个员工都有一个所属部门，假如有一个员工信息表 Employee(emp_id,emp_name,emp_age,dept_id,dept_

name,dept_info)。这张员工信息表的属性是 emp_id，因为这个属性能够唯一确定其他所有属性，比如知道员工编号 emp_id 以后，肯定能够知道员工姓名，所属部门编号，部门名称和部门介绍。所以这里 dept_id 不是主属性，而是非主属性。但是，我们又可以发现 dept_name,dept_info 这两个属性也可以由 dept_id 这个非主属性决定，即 dept_name 依赖 dept_id，而 dept_id 依赖 emp_id，这样就存在了传递依赖。而且我们可以看出传递依赖的一个明显缺点就是数据冗余非常严重。

那么如何解决传递依赖问题，其实非常简单，我们只需要将 dept_name,dept_info 这连个属性删除就可以了，即 Employee(emp_id,emp_name,emp_age,dept_id)，然后再创建一个部门表 Dept(dept_id,dept_name,dept_info)。这样如果要搜索某一个员工的部门信息 dept_info，可以通过数据库连接来实现，查询语句如下：

```
select e.emp_id,e.emp_name,d.dept_name from
Employee e,Dept d where e.dept_id=d.dept_id
```

BC 范式

设关系模式 R<U，F>∈1NF，如果对于 R 的每个函数依赖 X→Y，若 Y 不属于 X，则 X 必含有候选码，那么 R∈BCNF。

解释一下：对于关系模式 R，若 R 中的所有非平凡的、完全的函数依赖的决定因素是码，则 R 属于 BCNF。

若 R∈BCNF

每一个决定属性集（因素）都包含（候选）码

R 中的所有属性（主，非主属性）都完全函数依赖于码

R∈3NF（证明）

若 R∈3NF 则 R 不一定∈BCNF

在关系模式 STJ（S，T，J）中，S 表示学生，T 表示教师，J 表示课程。

每一教师只教一门课。每门课由一名教师教，某一学生选定某门课，就确定了一个固定的教师。某个学生选修某个教师的课就确定了所选课的名称 ： (S，J)→T，(S，T)→J，T→J

由关系模式的定义可以得到如下结论，若 R 属于 BCNF，则 R 有：

1.所有非主属性对每一个码都是完全函数依赖。

2.所有的主属性对每一个不包含它的码,也是完全函数依赖。

3.没有任何属性完全函数依赖于非码的任何一组属性。

由于 R∈BCNF,按定义排除了任何属性对码的传递依赖与部分依赖,所以 R∈3NF。但是若 R∈3NF,则 R 未必属于 BCNF。

1.第一范式：数据库的字段是单一属性，不可再分。

解释：

- 不能是复合属性，如果存在，应该拆分为多个属性
- 不能是多值属性，如果存在，应该建立一个实体，而让此属性与其存在 1 对多的关系)
- 不能是重复属性

2.第二范式：任何非关键字段不能部分依赖任一侯选关键字（即必须完全依赖）。

解释：

- 表中必须存在侯选关键字，即每一行不同于其他任一行，是惟一区分的
- 任何非关键字段不能依赖于侯选关键字的一部分

3.第三范式：任何非关键字段不能传递依赖任一侯选关键字

解释：

- 非关键字字段必须直接依赖任一侯选关键字
- 非关键字段 C 不能依赖非侯选关键字 B，因为样会形成传递依赖：侯选关键字 A=>B=>C，因为这时的 B 往往是外键，即其他表的主键，也就是说表中不能含有其他表的非主属性

4.BC 范式：任何字段都不能传递依赖任一侯选关键字

解释：

- 与第三范式相比，一个是“任何非关键字段不能”，一个是“任何字段不能”，显然更严格了
- 侯选关键字或其部分字段不能传递依赖其他的侯选关关键字

注释：

侯选关键字：又叫侯选码，惟一标识一行数据，其真子集不能是侯选关键字，一个表可以存在多个侯选关键字，如用户表的 username，userid

主关键字：又叫主键，主码，被选中的用来区分其它行的侯选关键字，一个表只有一个主关键字

部分依赖：(A,B)->C,D,如 A->C,则 C 部分依赖 A

传递依赖：A->B->C,则 C 传递依赖 A

1. 第一范式（1NF）：属性不可拆分 或 无重复的列

这个简单，就是一个属性不允许再分成多个属性来建立列。事实上，在目前的 DBMS 中是不可能拆分属性的，因为他们不允许这么做。

2. 第二范式（2NF）：完全函数依赖

先讲讲什么是部分函数依赖。

部分函数依赖，就是多个属性决定另一个属性，但事实上，这多个属性是有冗余的。例如，（学号，班级）->姓名，事实上，只需要学号就能决定姓名，因此班级是冗余的，应该去掉。

满足第二范式的数据库设计必须先满足第一范式。

因此第二范式的目标就是消除函数依赖关系中左边存在的冗余属性。

3.第三范式（3NF）：消除传递依赖

不依赖于其他非主属性（消除传递依赖）。

满足第三范式的数据库必须先满足第二范式。

也就是，数据库中的属性依赖仅能依赖于主属性，不存在于其他非主属性的关联。

例如，图书，图书室的关系。图书包括编号、出版商、页码等信息，图书室包括图书室编号、所存图书（外键）。其中，图书室的表中不应该存储任何图书的具体信息（例如，出版商。。），而只能通过主键图书编号来获得对应图书的信息。

4.BC 范式（BCNF）：

（1）所有非主属性对每一个码都是完全函数依赖；

（2）所有的主属性对于每一个不包含它的码，也是完全函数依赖；

（3）没有任何属性完全函数依赖于非码的任意一个组合。

R 属于 3NF，不一定属于 BCNF，如果 R 属于 BCNF，一定属于 3NF。

Boyce-Codd 范式（英语：**Boyce-Codd normal form**，缩写 **BCNF**），是数据库规范化中所使用的一种正规形式。是在第三范式的基础上加上更严格约束，每个 BCNF 关系是第三范式的子集，有从属关系。它的定义是：

> 如果对于关系模式 R 中存在的任意一个非平凡函数依赖 X->A，都满足 X 是 R 的一个候选键，那么关系模式 R 就属于 BCNF，。

BCNF 与第三范式的不同之处在于，第三范式允许 A 是主属性（第三范式中不存在非主键被另一个非主键决定），而在 BCNF 中，任何属性（包括非主键和主键）都不能被非主键所决定。

范例[编辑]

关系模式 R：

***Property_id*#(主键)**	**County_name**	**Lot#**	**Area**

其中依赖关系如下： Property_id#->{County_name,Lot#,Area}; {County_name,Lot#}->{Property_id#,Area}; Area->County_name;

很明显最后一个依赖违反了 BC 范式的要求，Area 不是关系模式 R 的主键，而依赖于它的 County_name 是能够决定其他属性的主属性。故应当规范化为：

***Property_id*#(主键)**	**County_name**	**Lot#**

***Area*(主键)**	**County_name**