1. 多表关系
2. 概述

数据库是由多个表组成的，表中存储的数据相互之间有着现实的意义。比如说学生管理系统中，我们设置来了一个学生表和一个课程表，一个学生可以选择多门课程，一门课程也可以被多个学生选择，这就是表之间的多对多的关系。

1. 多表之间的关系
2. 一对一

一种数据对应另一种数据的关系。如一个人只有一个身份证号码，而一个身份证号码只能被一个人所拥有。

1. 一对多（多对一）

如一个学生属于一个班级，而一个班级中有多个学生。

1. 多对多

学生与课程的关系。

1. 多表关系的实现
2. 一对一

在两个表中的任意一方中添加唯一的外键指向另一表的主键。注意，唯一的外键指的是该外键所在字段约束为UNIQUE，对应的另一表中的主键也是唯一的，一对一的关系成立。

1. 一对多

一个表A中的某个数据可以选择另一表B的多个数据，则选择表A的主键，因为表A只能选择一个数据，是唯一的；在表B中建立外键，因为要和表A建立联系，将表B的外键指向表A的主键。

1. 多对多

多对多的关系要借助第三张中间表，这个中间表要和两张表产生联系，就要在中间表中建立外键，且至少两个外键，对应着两个表，这两个外键分别指向两个表中的主键。

1. 数据库的设计
2. 范式

数据库的设计直接影响着数据的存储性能和后期的程序开发，要建立科学，规范的数据库就需要用规范的规则来优化数据库的设计，这些规则称为范式。

1. 范式的分类

目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF） 、

第四范式(4NF）和第五范式（5NF，又称完美范式）。

满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多规范要求的称为第二范式（2NF） ，

其余范式以次类推。一般说来，数据库只需满足第三范式(3NF）就行了。

1. 第一范式
2. 概述

数据库表的每一列都是不可分割的原子数据项，若某个列有多个值时i，需要拆分为不同的列，即第一范式的每一列都不可再拆分。

1. 隐患

第一范式是最基础的规则，有很多隐患：

1. 数据冗余非常严重，每个数据都会有大量的重复。
2. 数据在添加和删除时存在问题。
3. 第二范式
4. 基础概念
5. 函数依赖

A 🡪 B，如果通过A属性（属性组）的值，可以确定唯一B属性的值，则称B依赖于A。

如：学号 🡪姓名，（学号，课程名称）🡪分数

1. 完全函数依赖

A 🡪B，A是一个属性组，B属性的确定需要依赖于A属性组中的所有属性。

如：（学号，课程） 🡪分数

1. 部分函数依赖

A🡪B，A是一个属性组，B属性的确定只需要依赖于A属性组中的部分值。

如：（学号，课程）🡪姓名

1. 传递函数依赖

A🡪B ，B🡪C，通过A属性（属性组）的值，可以唯一确定B属性的值，通过B属性（属性组）的值，可以唯一确定C属性的值，则称C传递函数依赖于A。

如：学号🡪系名🡪系主任

1. 码

在一张表中，一个属性或属性组被其他所有属性完全依赖，即依靠这个属性（属性组）可以唯一确定其他所有属性，则称这个属性（属性组）为码。

主属性：码属性组中的所有属性

非主属性：除去码属性组的属性

1. 第二范式概述

在第一范式的基础上，非主属性必须完全依赖于主属性，不能存在仅依赖主键一部分的列。这样，主属性所对应的数据就不会重复，解决了数据的冗余问题。即每张表的非主属性和主属性的关系十分密切，有主属性能够直接唯一确定所有的非主属性，因此每张表只能描述一件事情。

如：设置一个借书证表：

学生证号 学生证名称 学生证办理时间 借书证号 借书证名称 借书证办理时间

主属性组：学生证号，借书证号

可以看出，现在的表是存在部分依赖的，将表分为两个表：

学生证号 学生证名称 学生证办理时间

借书证号 借书证名称 借书证办理时间

每个表都是完全依赖，不存在部分依赖问题，非主属性可以根据主属性推断出来，则主属性的数据不用重复了，数据冗余问题解决了。

1. 第三范式

在第二范式的基础上，任何非主属性不依赖与其他的非主属性，即每个非主属性都直接依赖于主属性，而不是通过其他的非主属性来传递依赖于主属性，消除传递依赖。

一般将有依赖关系的非主属性单独设置到另一张表中，再用外键和原表相关联起来。这样，每个表中的数据封闭性很强，修改某个表的数据对于其他表没有影响。

如：学生信息表

学号 姓名 年龄 所在学院 学院地点

主属性：学号

非主属性中存在传递依赖关系，学院和学院地点。因此，需要将学院和学院地点设置到另一张表中，并用外键和原表关联起来。

学号 姓名 年龄 所在学院的编号(外键)

学院编号 所在学院 学院地点