**范式、视图与触发器**

**一、设计数据库的三大范式**

关系数据库中的关系必须满足一定的要求。满足不同程度要求的为不同范式。数据库的设计范式是数据库设计所需要满足的规范。只有理解数据库的设计范式，才能设计出高效率、优雅的数据库，否则可能会设计出错误的数据库。

目前，主要有六种范式：第一范式、第二范式、第三范式、BC范式、第四范式和第五范式。满足最低要求的叫第一范式，简称1NF。在第一范式基础上进一步满足一些要求的为第二范式，简称2NF。其余依此类推。

范式可以避免数据冗余，减少数据库的空间，减轻维护数据完整性的麻烦，但是操作困难，因为需要联系多个表才能得到所需要数据，而且范式越高性能就会越差。要权衡是否使用更高范式是比较麻烦的，一般在项目中，用得最多的也就是第三范式，我认为使用到第三范式也就足够了，性能好而且方便管理数据。

函数依赖:如果一个表中某一个字段Y的值是由另外一个字段或一组字段X的值来确定的，就称为Y函数依赖于X。  
**第一范式(1NF)**  
定义：如果关系模式R的每个关系r的属性都是不可分的数据项，那么就称R是第一范式的模式。  
简单的说，每一个属性都是原子项，不可分割。  
1NF是关系模式应具备的最起码的条件，如果数据库设计不能满足第一范式，就不称为关系型数据库。关系数据库设计研究的关系规范化是在1NF之上进行的。

例如(学生信息表)：  
学生编号 姓名 性别 联系方式  
20080901 张三 男 email:zs@126.com,phone:88886666  
20080902 李四 女 email:ls@126.com,phone:66668888

以上的表就不符合，第一范式：联系方式字段可以再分，所以变更为正确的是：

学生编号 姓名 性别 电子邮件 电话  
20080901 张三 男 zs@126.com 88886666  
20080902 李四 女 ls@126.com 66668888

**第二范式（2NF）**  
定义：如果关系模式R是1NF，且每个非主属性完全函数依赖于候选键，那么就称R是第二范式。  
简单的说，第二范式要满足以下的条件：首先要满足第一范式，其次每个非主属性要完全函数依赖与候选键，或者是主键。也就是说，每个非主属性是由整个主键函数决定的，而不能由主键的一部分来决定。

例如(学生选课表):  
学生 课程 教师 教师职称 教材 教室 上课时间  
李四 Spring 张老师 java讲师 《Spring深入浅出》 ３０１ 08:00  
张三 Struts 杨老师 java讲师 《Struts in Action》 ３０２ 13:30

这里通过（学生，课程）可以确定教师、教师职称，教材，教室和上课时间，所以可以把（学生，课程）作为主键。但是，教材并不完全依赖于（学生，课程），只拿出课程就可以确定教材，因为一个课程，一定指定了某个教材。这就叫不完全依赖，或者部分依赖。出现这种情况，就不满足第二范式。

修改后，选课表：  
学生 课程 教师 教师职称 教室 上课时间  
李四 Spring 张老师 java讲师 ３０１ 08:00  
张三 Struts 杨老师 java讲师 ３０２ 13:30

课程表：  
课程 教材   
Spring 《Spring深入浅出》   
Struts 《Struts in Action》

所以，第二范式可以说是消除部分依赖。第二范式可以减少插入异常，删除异常和修改异常。

**第三范式（3NF）**  
定义：如果关系模式R是2NF，且关系模式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依赖，则称关系R是属于第三范式。    
简单的说，第三范式要满足以下的条件：首先要满足第二范式，其次非主属性之间不存在函数依赖。由于满足了第二范式，表示每个非主属性都函数依赖于主键。如果非主属性之间存在了函数依赖，就会存在传递依赖，这样就不满足第三范式。

上例中修改后的选课表中，一个教师能确定一个教师职称。这样，教师依赖于（学生，课程），而教师职称又依赖于教师，这叫传递依赖。第三范式就是要消除传递依赖。

修改后，选课表：

学生 课程 教师 教室 上课时间  
李四 Spring 张老师 ３０１ 08:00  
张三 Struts 杨老师 ３０２ 13:30

教师表：  
教师 教师职称  
张老师 java讲师  
杨老师 java讲师

这样，新教师的职称在没被选课的时候也有地方存了，没人选这个教师的课的时候教师的职称也不至于被删除，修改教师职称时只修改教师表就可以了。

简单的说，  
第一范式就是原子性，字段不可再分割；  
第二范式就是完全依赖，没有部分依赖；  
第三范式就是没有传递依赖。

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。这并不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 比如 一对一 或一对多关系的这种情况下，合并导致的不符合范式要求反而是合理的。

**二、视图**

**1、理解视图**

视图与表有很多相似的地方，视图也是由若干字段以及若干记录构成的，它也可以作为select语句的数据源。甚至在某些特定情况下，可以通过视图对表进行更新操作。

视图保存的仅仅是一条select语句，其源数据都来自于数据库表，数据库表称为**基本表或者基表**，视图称称为**虚表**。**基表数据发生变化，虚表也会随之变化。**

**2、视图作用**

1.使操作变得简单（可以将复杂的select语句封装到视图）

2.避免数据冗余（可以由一个基表或者多个基表产生多种针对性的视图）

3.增强数据安全性（为不同的用户分配不通的视图，增强了数据的安全访问控制）

4.提高数据的逻辑独立性（使应用程序与数据逻辑独立）

**3、创建视图**

语法：

create view 视图名[(视图字段列表)]

as select语句

为了区分视图与基本表，在命名视图时，建议加前缀“view\_”或者“v\_”

例如：

/\*将select子查询的结果集作为视图的数据源\*/

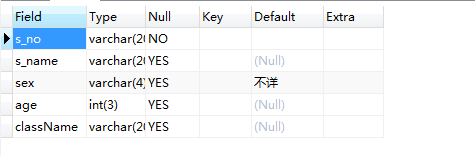
create view v\_student\_info as select students.s\_no,students.s\_name,students.sex,students.age,class.className from students left join class on students.class\_id=class.id;

create view v\_teacher\_info as select s\_teacher.s\_no,s\_teacher.s\_name,s\_teacher.s\_sex from s\_teacher where s\_sex=1;

**4、查看视图**

describe 视图名

例如：describe v\_student\_info;



**5、删除视图**

drop view 视图名

**三、触发器**

**1、理解触发器**

mysql5.0新增的功能，触发器定义了一系列的操作，这一系列的操作称为触发程序，当触发事件发生时，触发程序自动运行。

**2、作用**

触发器主要用于监视某个表的insert、update、delete等更新操作，这些操作可以分别激活该表的insert、update或者delete类的触发程序运行，从而实现数据自动维护。

**3、创建触发器**

create trigger 触发器名 触发时机 触发事件 on 表名 for each row

begin

触发程序

end;

触发时机：before与after

触发事件：insert、update、delete

for each row表示行级触发器

使用old和new关键字对记录进行操作。new.字段名或者old.字段名。 old记录是只读的。new记录只有是before类型时才能更改值

例如：

**Insert类触发器(before)**

/\*new代表新数据创建时使用，创建一个触发器当插入之前学员数据年龄小于18的，就自动默认18岁\*/

create trigger tr\_student before insert on students for each row

begin

if(new.age<18)

then

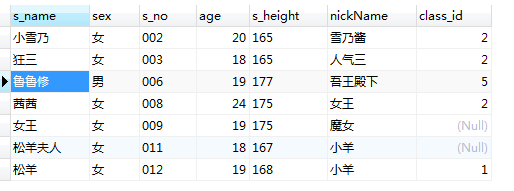
set new.age = 18;

end if;

end

insert into students(s\_no,s\_name,age,sex,s\_height,nickName,class\_id) values('011','松羊夫人',17,'女',167,'小羊',default);

insert into students(s\_no,s\_name,age,sex,s\_height,nickName,class\_id) values('012','松羊',19,'女',168,'小羊',1);



**Update类触发器(before)**

/\*old代表在更新数据之前的原先数据，更新数据时，如果年龄少于18，就自动默认18，身高小于165，就用原先的身高\*/

create trigger tr\_student\_update before update on students for each row

begin

if(new.age<18)

then

set new.age = 18;

end if;

if(new.s\_height<165)

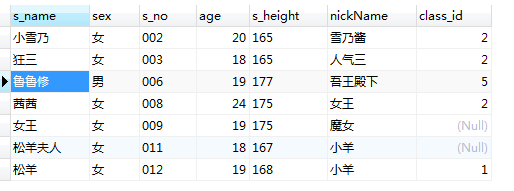
then

set new.s\_height = old.s\_height;

end if;

end

update students set s\_height=164 where s\_no='006';



**Delete类触发器****(before)**

/\*删除表名为test的高级班数据，同时删除class表的数据\*/

create trigger tr\_test\_delete before delete on test for each row

begin

/\*delete触发器中不能够有new关键字\*/

delete from class where className=old.className;

end;

delete from test where className='高级班';

**Insert类触发器(after)**

例如：

/\*当students表插入新数据后，触发器同时给students\_bak表插入新的数据\*/

create trigger tr\_student\_after after insert on students for each row

begin

insert into students\_bak values(new.s\_name,new.sex,new.s\_no,new.age);

end

insert into students(s\_no,s\_name,age,sex,s\_height,nickName,class\_id) values('013','松松',15,'女',168,'小松',1);

**Update类触发器(after)**

例如：

/\*更新students表数据的同时更新students\_bak表的数据\*/

create trigger tr\_student\_after\_update after update on students for each row

begin

update students\_bak set s\_name=new.s\_name where s\_name='黑黝黝';

end

update students set s\_name='heiyouyou' where s\_no='007';

**Delete类触发器(after)**

/\*删除students表的数据后，同时删除另一个表students\_bak的相同数据\*/

create trigger tr\_student\_after\_delete after delete on students for each row

begin

delete from students\_bak where s\_name=old.s\_name;

end

delete from students where s\_name='松松';

**4、查看触发器的定义**

show create trigger 触发器名称

例如：

show create trigger tr\_student\_after\_delete



**5、删除触发器**

drop trigger 触发器名

drop trigger tr\_student\_after\_delete

