链滴

数据库设计方法论 - 继承

作者：jianzh5

原文链接：https://ld246.com/article/1571650102107

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20190825.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
继承这个概念做 java 开发的同学应该都很熟悉了，继承指的是子类继承父类的特征和行为，使
子类对象（实例）具有父类的实例域和方法，或子类从父类继承方法，使得子类具有父类相同的行为

数据库设计的时候也是有继承关系的，在数据库设计方法论中继承有三种,分别是具体表继承（Concre
e Table Inheritance）、单表继承（Single Table Inheritance）、类表继承（Class Table Inheritan
e）。我们实际设计中经常会不经意中使用到数据库到继承，下面分别介绍一下他们的概念：
<h2 id="概念解析">概念解析</h2>

具体表继承
不建立父对象，将父对象的所有属性转移到子对象中，为每个子对象建立对于的表。

单表继承
在一个宽表中列出所有父对象和子对象的属性，同时用一个标识列表示该行数据存储的是哪个子类的
据。

类表继承
对父对象和每个子对象建立一个对应的表，然后在子表中设置该子表的主键为与父表关联的外键。</
>

<h2 id="设计示例">设计示例</h2>
假如你现在在做个教学系统，系统中有三个角色：学生、家长、老师。
学生的属性：姓名、年龄、性别、身份证、入学时间、学号、学分
家长的属性：姓名、年龄、性别、职业、学历
老师的属性：姓名、年龄、性别、教龄、学科、是否已婚
不同继承方案的实现如下：

具体表继承(三张独立表)

学生表(ID、姓名、年龄、性别、身份证、入学时间、学号、学分)
家长表(ID、姓名、年龄、性别、职业、学历)
老师表(ID、姓名、年龄、性别、教龄、学科、是否已婚)

单表继承（一张大宽表 + 类型字段用以区分）

用户表(ID、姓名、年龄、性别、身份证、入学时间、学号、学分、职业、学历、教龄、学科、
否已婚、类型)

类表继承(四张表：一张父表表 + 三张子表)

用户表(ID、姓名、年龄、性别)
学生表(ID、用户 ID、身份证、入学时间、学号、学分)
家长表(ID、用户 ID、职业、学历)
老师表(ID、用户 ID、教龄、学科、是否已婚)

<h2 id="方案对比">方案对比</h2>

具体表继承
优点：获取完整对象不需要联表查询；表中没有无关属性（跟单表继承的对比）
缺点：添加公共属性时需要修改多个表；查询公共字段展示需要查询多个表并作 <code>union<code> 操作(如:页面需要展示所有的用户，显示用户的公共字段)
使用场景：适用于子表关联性较弱的业务场景，并且识别出系统没有公共数据查询的求

单表继承
优点：库表设计简单，获取子表数据时不需要 join 连接。
缺点：表空间利用率低，子表出现无关属性；扩展子表属性时需要修改数据表(锁表)。
使用场景：　适用于子类属性较少的情况。比如可预见的时间内子类的属性都比较少可以使用这种方式，毕竟查询简单，不需要联表查询。

类表继承
优点：库表的层次结构清晰；为子类添加属性不用修改父表，添加公共属性不需要修改子表；查公共数据时不需要去查询多个表；扩展性强
缺点：获取对象完整数据需要 join 查询,在数据量很大时影响查询效率
使用场景：适用关联性较强的业务场景，子表属性变化较大。

<h2 id="总结">总结</h2>
数据库设计的原则就是没有原则，需要根据业务场景选择具体的设计方法。

今天说的数据库继承方案或者数据库范式都是这样，并不是说你数据库设计的扩展性强或者完全循 4NF 范式消除一切数据一致性问题就最好，设计带来的 join 查询效率也需要慎重考虑。
<blockquote>
欢迎关注个人公众号：JAVA 日知录
</blockquote>