# 数据库设计四种主要设计模式—

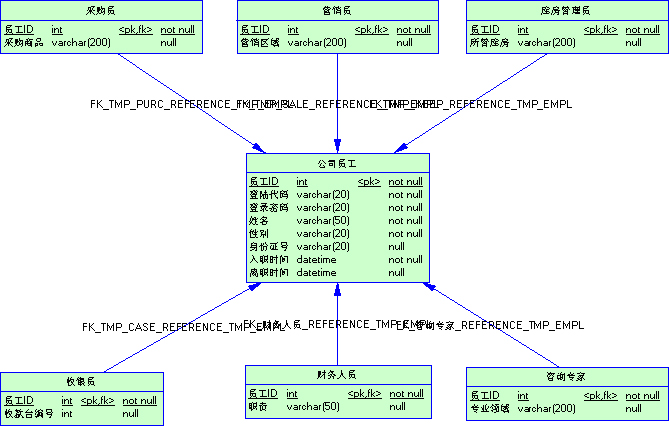
[**（一）主扩展模式**](http://blog.csdn.net/liu7537/article/details/657230)

主扩展模式，通常用来将几个相似的对象的共有属性抽取出来，形成一个“公共属性表”；其余属性则分别形成“专有属性表”，且“公共属性表”与“专有属性表”都是“一对一”的关系。

“专有属性表”可以看作是对“公共属性表”的扩展，两者合在一起就是对一个特定对象的完整描述，故此得名“主扩展模式”。

举例如下（注：这个例子已经作了相当程度的简化，仅仅是用来帮助大家理解“主扩展模式”这个概念来使用的，请大家注意）。

假设某公司包括如下6种类型的工作人员：采购员、营销员、库房管理员、收银员、财务人员和咨询专家，采用主扩展模式进行设计，如下图所示。



无论哪种类型的工作人员，都要访问公司的办公软件，所以都有“登陆代码”和“登录密码”；并且作为一般属性，“姓名”、“性别”、“身份证号”、“入职时间”、“离职时间”等属性，都与个人所从事的工作岗位无关，所以可以抽取出来作为公共属性，创建“公司员工”表。

很显然，公司委派员工采购哪些商品是“采购员”的专有属性，这是由公司的实际业务特点决定的。换句话说，公司不可能把采购任务放到“营销员”身上，也不可能放到“库房管理员”身上，“采购商品”属性就是“采购员”的专用属性。

“采购员”表的主键与“公司员工”表的主键是相同的，包括字段名称和字段的实际取值；“采购员”表的主键同时是“公司员工”表主键的外键。在PDM图里可以看到“采购员”表中的“员工ID”字段后面有一个“<pk,fk>”标记，这个标记就说明“员工ID”字段既是“采购员”表的主键，同时也是该表的外键。

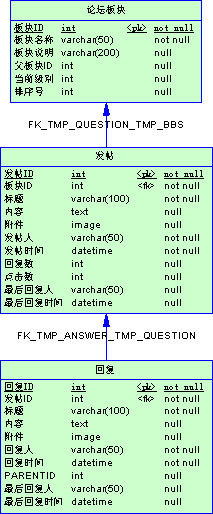
“公司员工”表是主表，“采购员”表是扩展表，二者是“一对一”的关系，两个表的字段合起来就是对“采购员”这个对象的完整说明。同理，“公司员工”表和其他5个表之间也都分别构成了“一对一”的关系。

对于主表来说，从表既可以没有记录，也可以有唯一一条记录来对主表进行扩展说明，这就是“主扩展模式”。

[**（二）主从模式**](http://blog.csdn.net/liu7537/article/details/657234)

主从模式，是数据库设计模式中最常见、也是大家日常设计工作中用的最多的一种模式，它描述了两个表之间的主从关系，是典型的“一对多”关系。

举例如下（注：这个例子已经作了相当程度的简化，仅仅是用来帮助大家理解“主从模式”这个概念来使用的，请大家注意）。

比如论坛程序。一个论坛通常都会有若干“板块”，在每个板块里面，大家可以发布很多的新帖。这时候“板块”和“发帖”就是主从模式，主表是“板块”，从表是“发帖”，二者是“一对多”的关系。多个潜水员也可以对感兴趣的同一份发帖进行回复，以表达各自的意见，这时候，一个“发帖”就有了多份“回复”，又构成了一个“主从模式”。

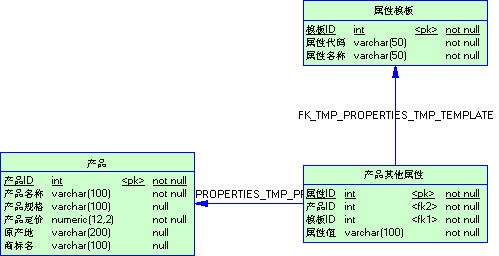
**（三）名值模式**

**名值模式，通常用来描述在系统设计阶段不能完全确定属性的对象，这些对象的属性在系统运行时会有很大的变更，或者是多个对象之间的属性存在很大的差异。**

举例如下（注：这个例子已经作了相当程度的简化，仅仅是用来帮助大家理解“名值模式”这个概念来使用的，请大家注意）。

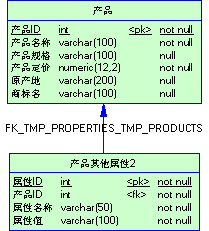
1.       使用名值模式进行设计时，如果对“其他属性”仅作浏览保存、不作其它任何特殊处理，则通常会设计一个“属性模板”表，该表的数据记录在系统运行时动态维护。

系统运行时，如需维护“产品其他属性”，可先从“属性模板”中选择一个属性名称，然后填写“属性值”保存，系统会将对应的产品ID、属性模板ID及刚刚填写的“属性值”一起保存在“产品其他属性”里，这样就完成了相关设置。无论产品的其他属性需求发生怎样的变化、怎样增删改属性，都可以在运行时实现，而不必修改数据库设计和程序代码。（见下图）



2.       使用名值模式进行设计时，如果对“其他属性”有特殊处理，比如统计汇总，那么这个属性名称需要在程序代码中作“硬编码”，即该属性名称需要在程序代码中有所体现，此时可以在“产品其他属性”表中直接记录“属性名称”，不再需要“属性模板”表。

系统运行时，如需维护“产品其他属性”，程序直接列出“属性名称”，然后填写“属性值”保存，系统会将对应的产品ID、属性名称及刚刚填写的“属性值”一起保存在“产品其他属性”里，这样就完成了相关设置。以后如果需求发生变更，则只需修改相应的程序代码即可，不必修改数据库设计。（见下图）



（四）多对多模式

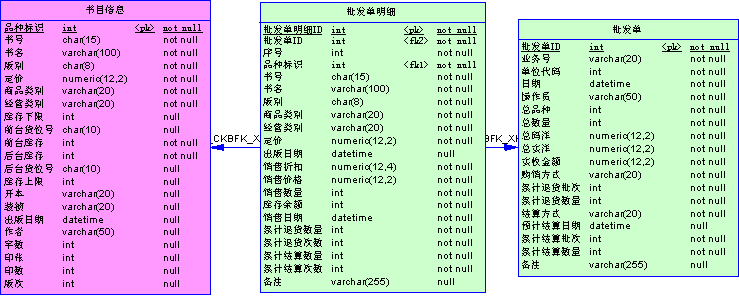
多对多模式，也是比较常见的一种数据库设计模式，它所描述的两个对象不分主次、地位对等、互为一对多的关系。对于A表来说，一条记录对应着B表的多条记录，反过来对于B表来说，一条记录也对应着A表的多条记录，这种情况就是“多对多模式”。

“多对多模式”需要在A表和B表之间有一个关联表，这个关联表也是“多对多模式”的核心所在。根据关联表是否有独立的业务处理需求，可将其划分为两种细分情况。

1.       关联表有独立的业务处理需求。

举例如下（注：这个例子已经作了相当程度的简化，仅仅是用来帮助大家理解“多对多模式”这个概念来使用的，请大家注意）。

比如网上书店，通常都会有“书目信息”和“批发单”。一条“书目信息”面对不同的购买客户、可以存在多张“批发单”，反过来，一张“批发单”也可以批发多条书目，这就是多对多模式。中间的“批发单明细”表就是两者的关联表，具备独立的业务处理需求，是一个业务实体对象，因此它具备一些特有的属性，比如针对每一条明细记录而言的“累计退货次数”、“累计退货数量”、“累计结算次数”、“累计结算数量”；由于批发单明细在数据产生后已经打印出纸质清单提供给客户，因此在“批发单明细”表里对纸质清单中打印的书目信息属性作了冗余（逆标准化），这样在将来即使修改了“书目信息”表中的属性，也不会影响跟客户核对批发单明细，不会影响未来的财务结算业务。



2.       关联表没有独立的业务处理需求

举例如下（注：这个例子已经作了相当程度的简化，仅仅是用来帮助大家理解“多对多模式”这个概念来使用的，请大家注意）。

比如用户与角色之间的关系，一般系统在做权限控制方面的程序时都会涉及到“系统用户表”和“系统角色表”。一个用户可以从属于多个角色，反过来一个角色里面也可以包含多个用户，两者也是典型的“多对多关系”。其中的关联表“用户角色关联表”在绝大多数情况下都是仅仅用作表示用户与角色之间的关联关系，本身不具备独立的业务处理需求，所以也就没有什么特殊的属性。

