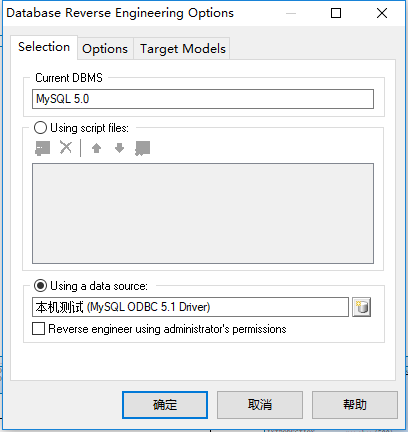
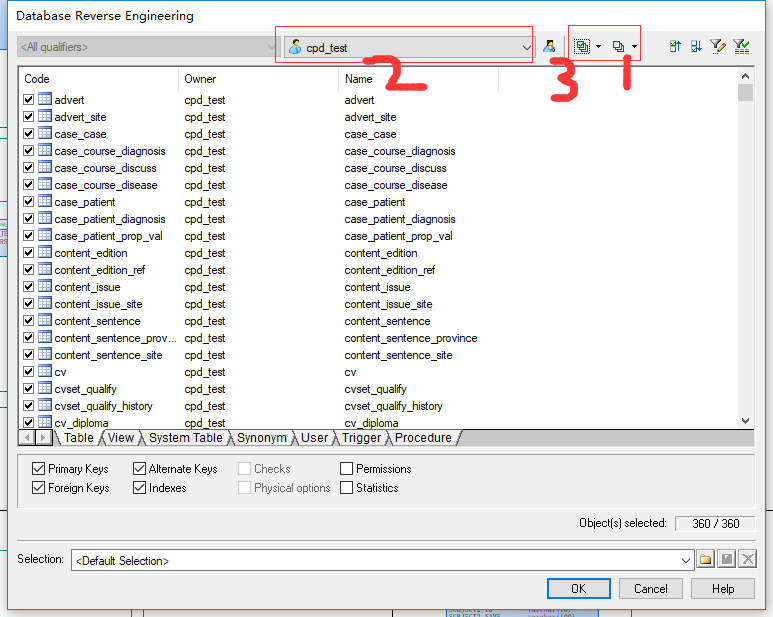
## PowerDesigner逆向工程从现有数据库生成PDM

### 1：没有PDM，首次拉取

配置了数据源的



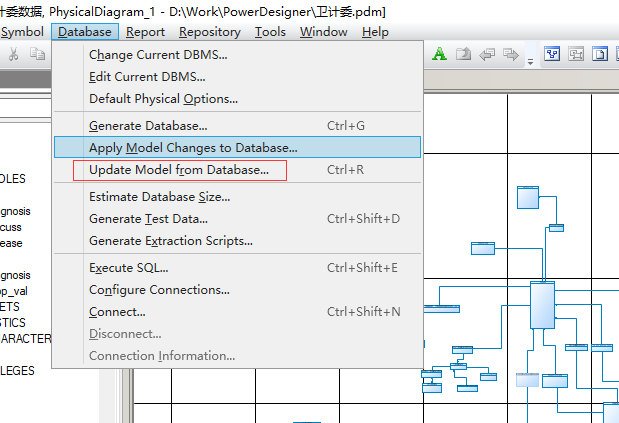
弹出提示框

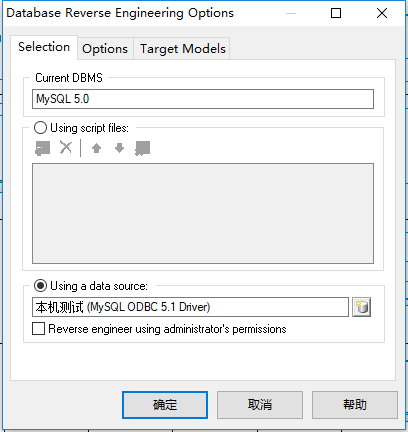


先取消选择，再选择对应的登陆账号，然后再全选。

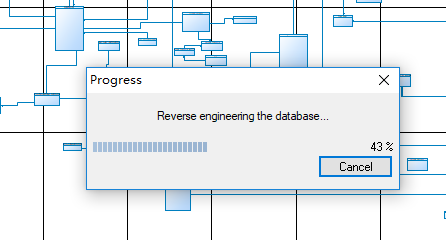
没有配置数据源的

### 2：已有PDM，再次更新

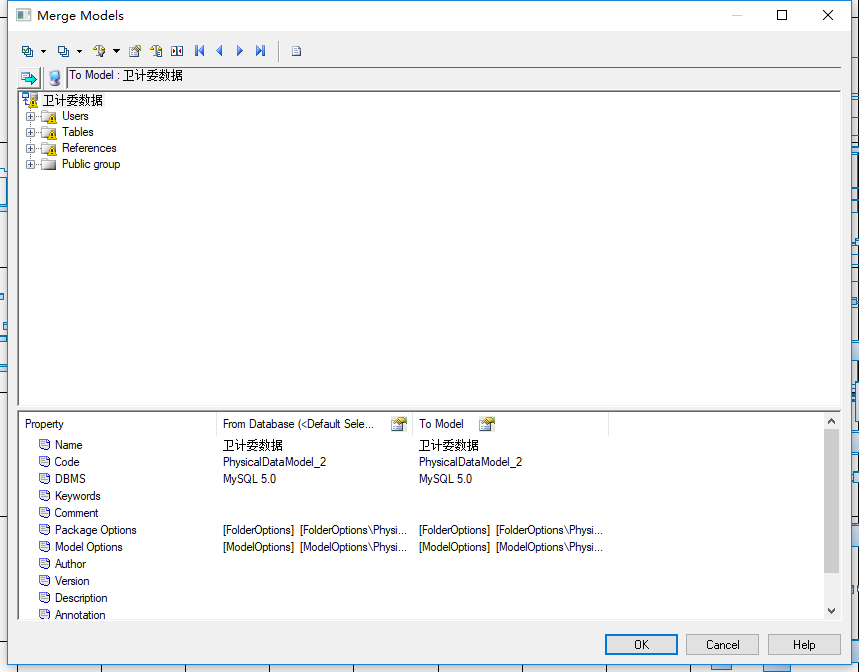




显示更新进程



显示合并选择框。



点击确定，执行完成，即可生成，



可能的标识。

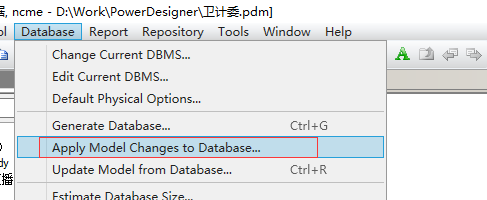
注释：

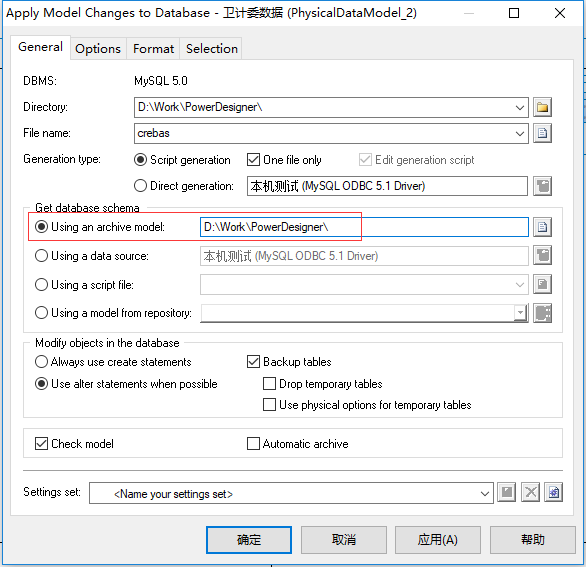
从数据库拉取更新到PDM的时候，如果，PDM已改修改，则该修改丢失（已发现：表中的注释）。

## 生成数据库文档

## powerdesigner导出后修改数据库结构数据库只进行更新的操作

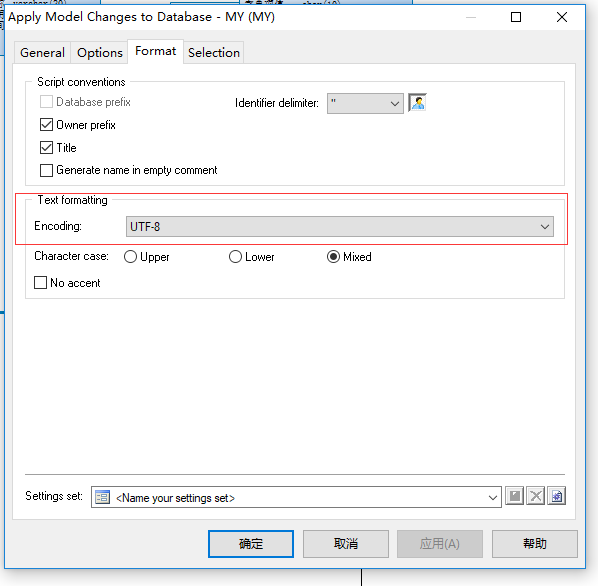
### PDM修改，将更新同步到数据库中。





目前是失败状态，待再次测试。

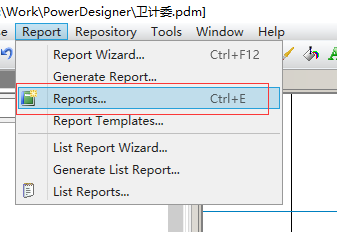
如果导入数据库的中文有乱码问题，请修改

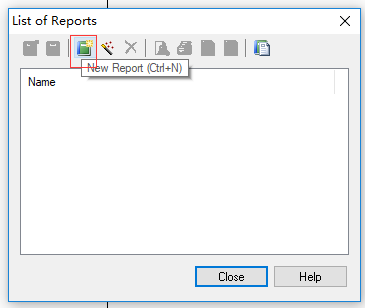


注意：

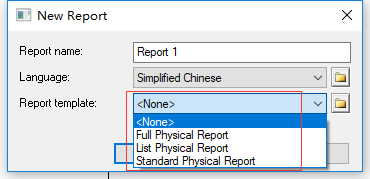
1：如果修改了表名，更新PDM时，会保留原表名，新建一个新表明的表。

## PDM的出为DOC

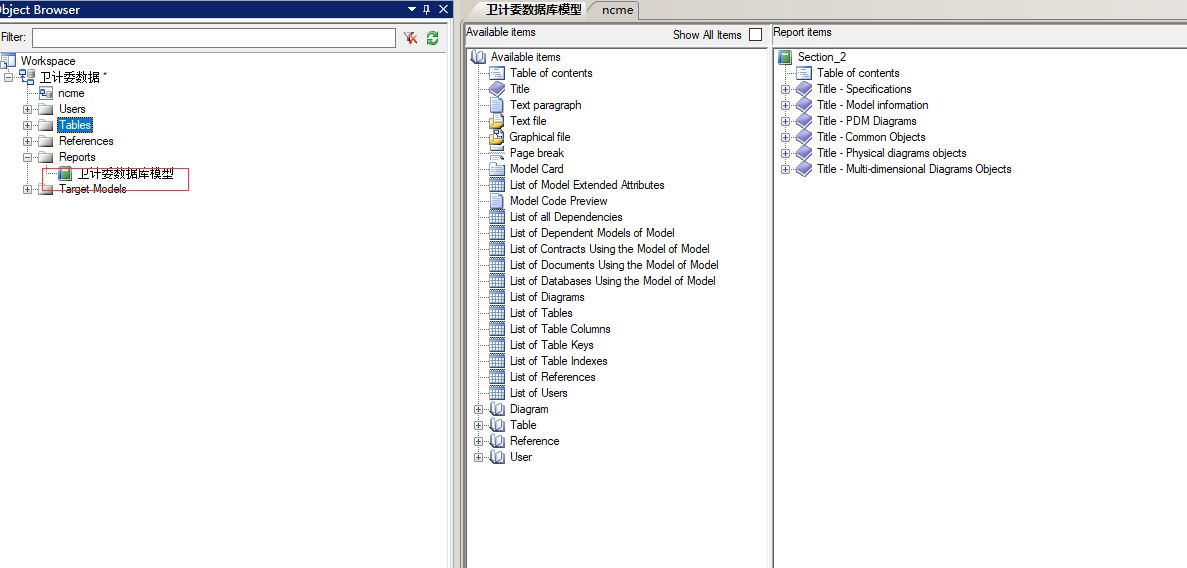




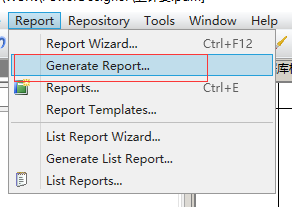
选择导入模板

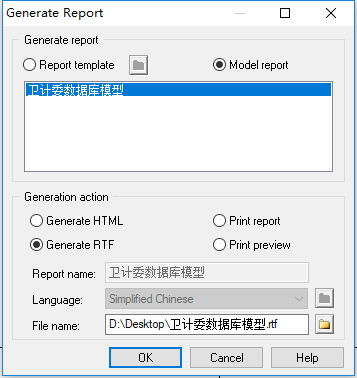


选择完模板，点击确定，即会生成，模型配置。

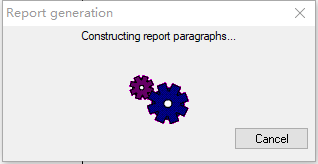


点击Report-->Generate Reports 进入导出页面

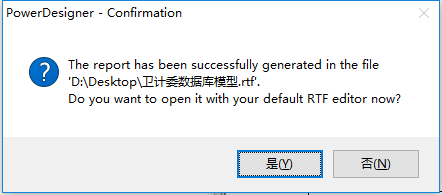




点击OK，生成的过程根据选择的模板和数据库本身大小确定。



弹出以下信息说明导出成功,打开查看导出文档

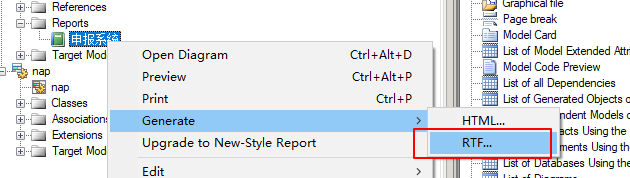


参考资料：文件中的wordTemplate.rtp模板。

系统默认的模板路径

C:\Program Files (x86)\Sybase\PowerDesigner 16\Resource Files\Report Templates

### 方法二：



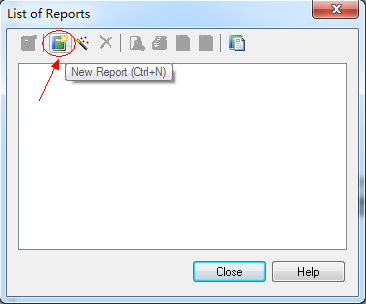
### 自定义导出模板

pd导出word表跟一般软件的导出功能类似，通过模版定义格式和样式，然后导出自定义格式的word。我用PowerDesigner15，导出功能用起来比较方便容易，唯一不爽的就是title page和前两页的Header and Footer取不掉。

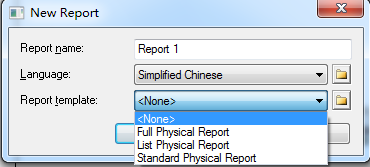
下面记下简单操作过程以备以后使用，[点击下载通用模版](http://download.csdn.net/detail/ferry_passion/5411353)

#### 一：基本使用介绍

１. 在工具栏中选择【Report -->Reports】，如下图



2. 点击第二个图标创建一个Report，如下图



该wizard中有三个信息

Report name Report : Report名字，也是导出word的文件名

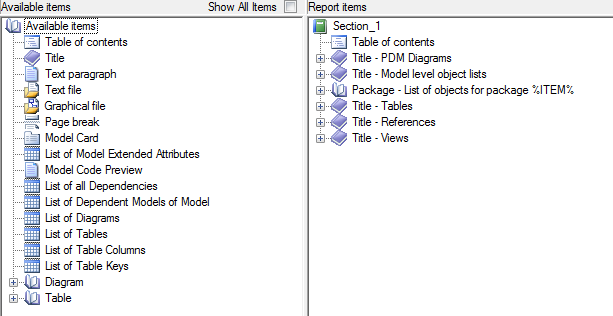
Language ：语言，有中文汉化包选择Simplified Chinese

Report template : Report模版，如果所示软件本身自带了几个模版，在安装目录Resource Files\Report Templates下

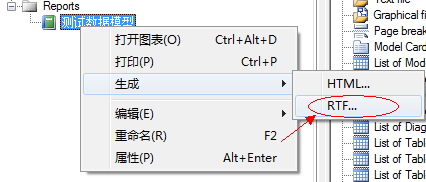
3. 选一个默认模版Standard Physical Report，确认后会发现多了两个区Available items和Report items

Available items ：软件提供的可用模块，是导出word模版的一个小组件，双击就会添加到Report items区域

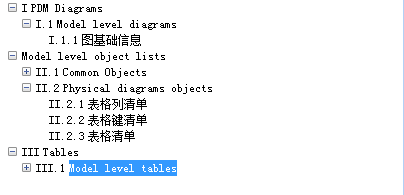
Report items : 我选中的模版小组件



4. 根据模版导出word，在工作空间区域，右键Reports下我们建的Report，如下图点击生产RTF格式



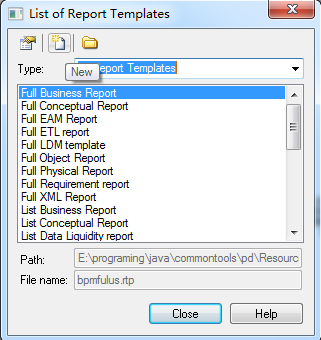
以为系统模版要导出的数据比较多会很慢，导出的word格式如下图所示



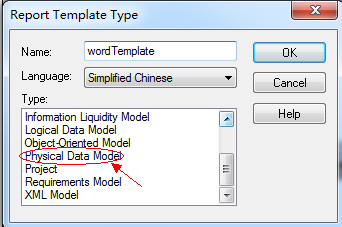
#### 二 ： 定制导出模版

当然这不是我们想要的word，下面看如何做一个自定义模版

1. 在工具栏中选择【Report -->Reports】，如下图

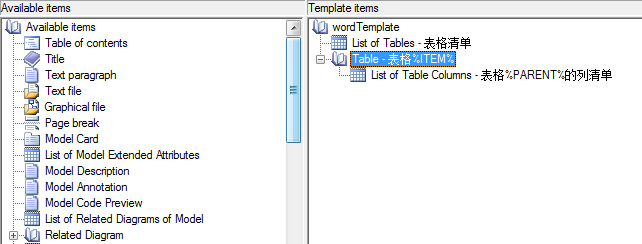


点击第二个图标创建一个Template，如下图



2. 选中Physical Data Model，OK确认

在Available items区域找到List of Tables和List of Table Columns，双击添加到Template items区域

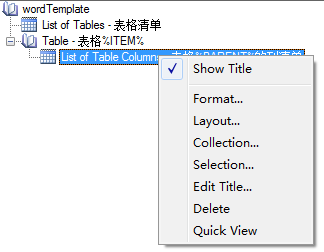


一般导出word也就只需要这两个

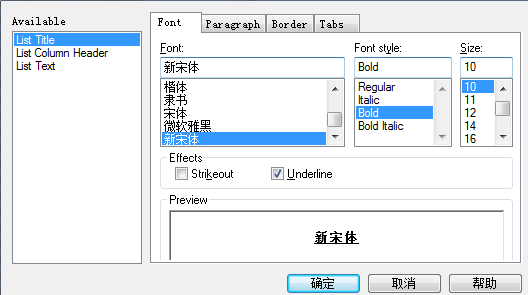
List of Tables ：所有表的清单，默认显示表名和中文描述，一般不需要改动

List of Table Columns ：表的列信息

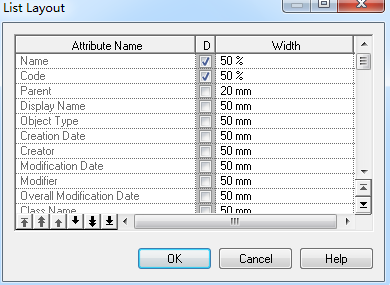
3. 右键List of Table Columns，如下图所示



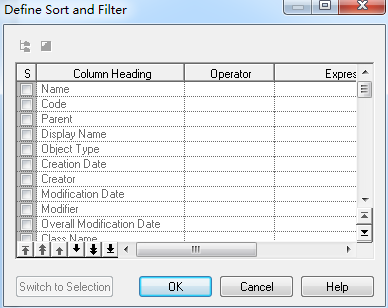
Format：字体设置，一般用默认的



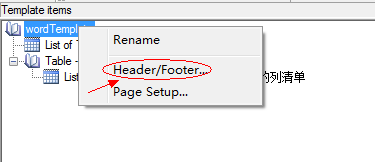
Layout：需要显示的属性和显示的宽度



Selection：排序和属性过滤，一般用默认的

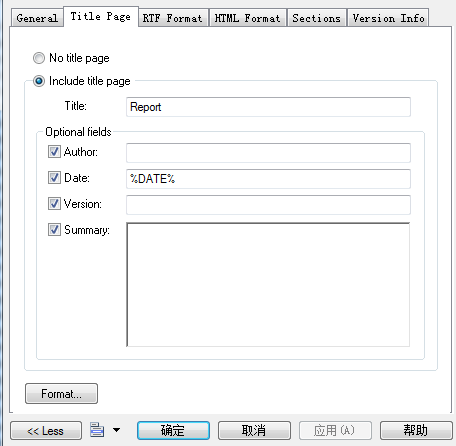


4. 设置导出word文档的页眉和页脚，右键模版如下图，不爽的是第一和第二页的页眉页脚这个设置不了



5. 设置title page，右键report如下图

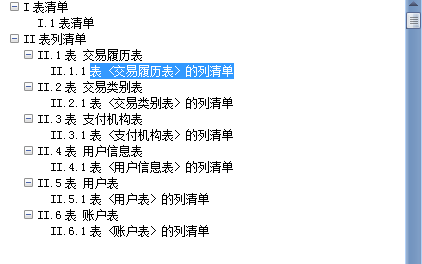




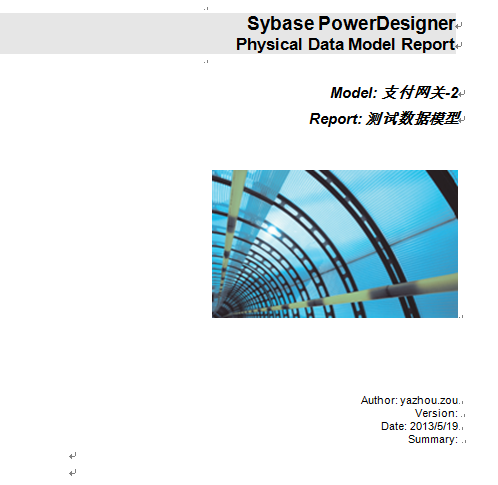
选了no title page竟然没效果，郁闷

6. 做了一个通用的模版，显示表清单和表列清单，[需要下载点击这里](http://download.csdn.net/detail/ferry_passion/5411353)，下面是导出后的效果图

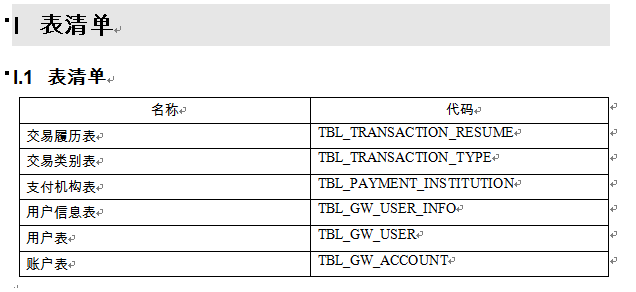
文档结构图：



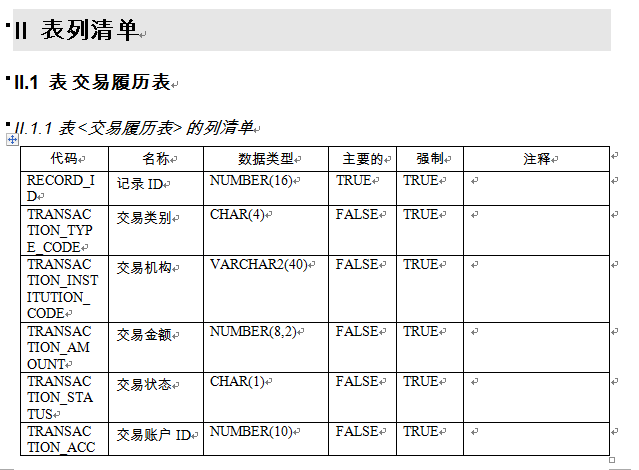
首页：



表清单：



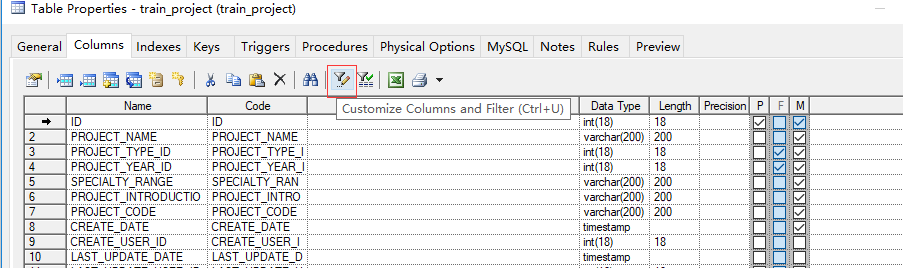
表列清单：



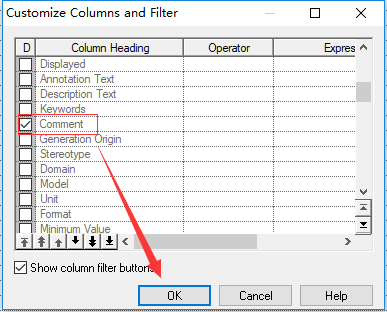
# ****使用技巧****

## ****显示comment列****

Table Properties（表属性）→ Columns（列）→ Customize Columns and Filter（自定义列过滤） 勾上 comment

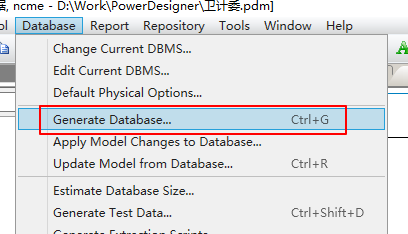


弹框中勾上comment，点击确定即可。

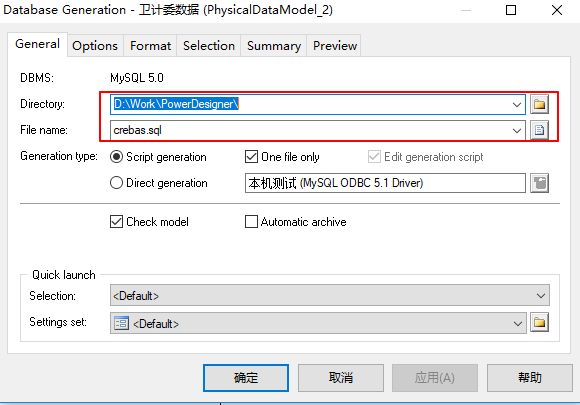


## 导出SQL 语句

DataBase(数据库) → Generate Database （生成数据库）→ confirm 确定



选择导出路径和保存文件名称。



本机测试失败，需要再次测试。

## 导出实体类

Tool（工具） → Generate Object-Oriented Model(生成适应模型) →

选择C# 确定 → Edit Current object language（编辑当前面向对象语言）

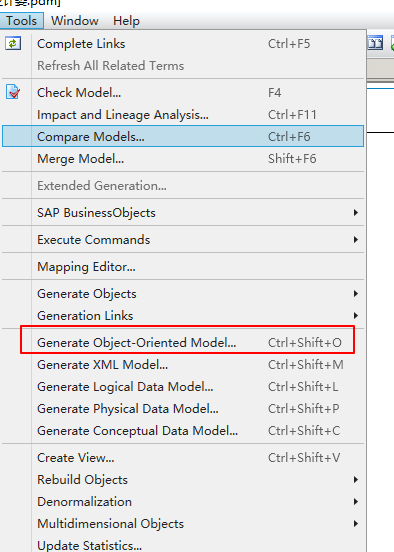
→ Profile\Attribute\Templates\definition

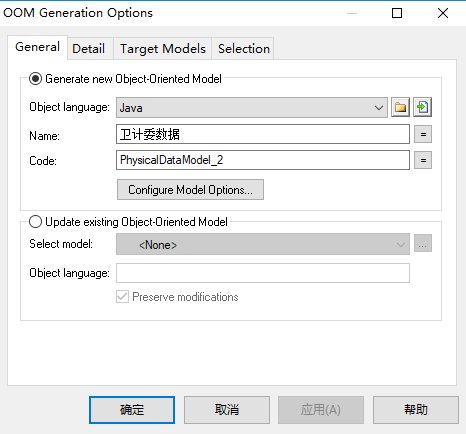
→ Profile\Classifier\Templates\DefaultHeader

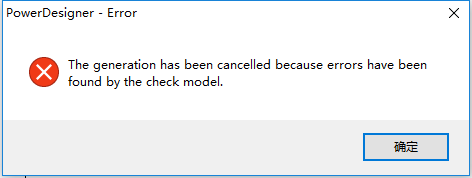
→ Profile\Shared\Templates\comment

→ Profile\Classifier\Templates\source

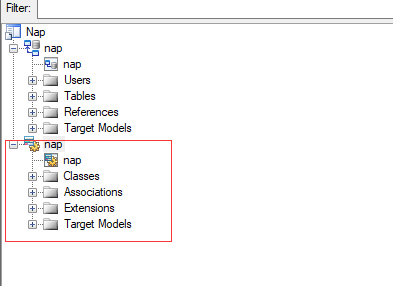
→ Generate C# Code



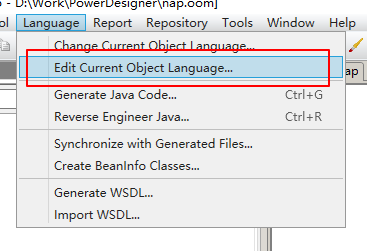




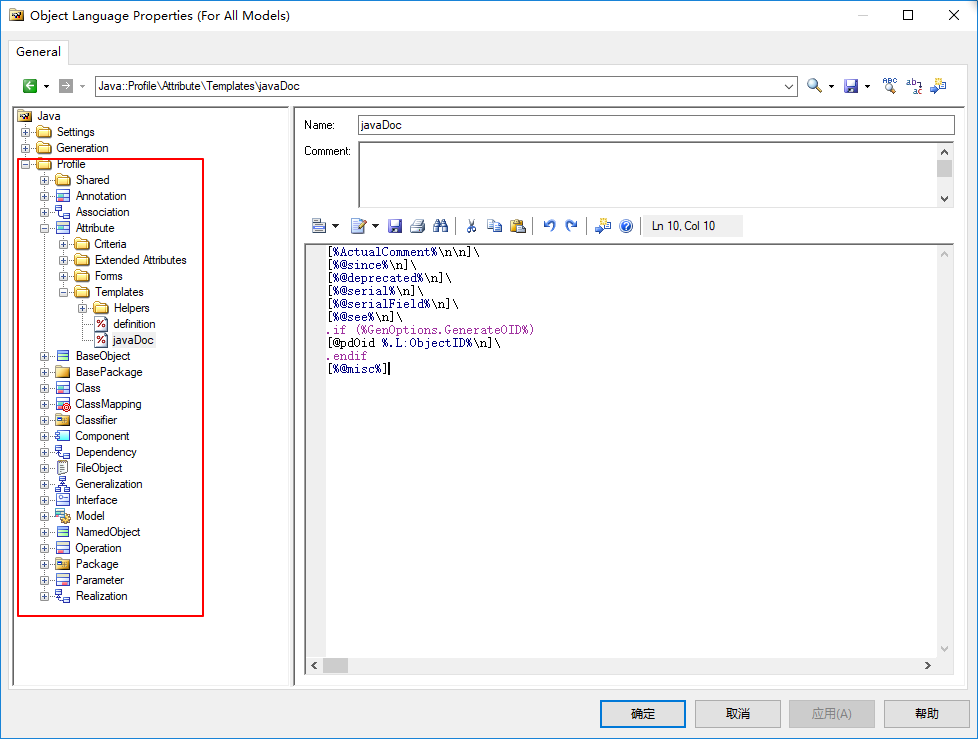
如果是正常的PDM会生成一个新的OOM文件



编辑语言类型

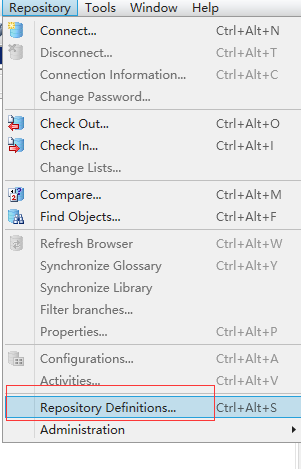


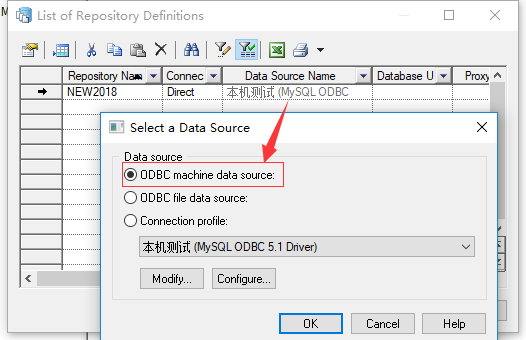
修改对应的导出模板



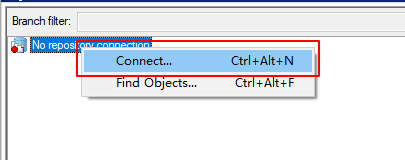
## PowerDesigner Repository建立模型库

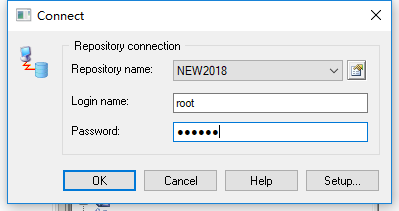
### **打开PowerDesigner，定义Repository库**

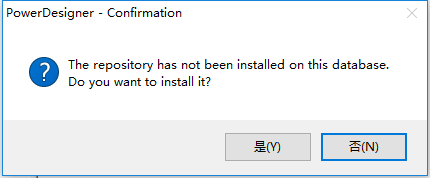




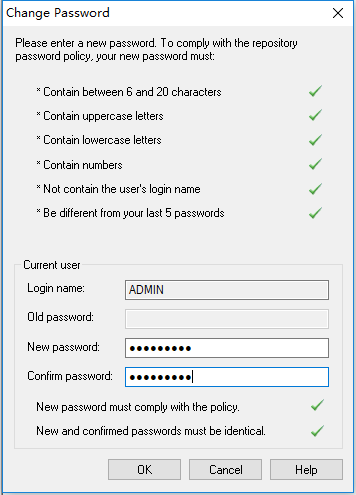
连接



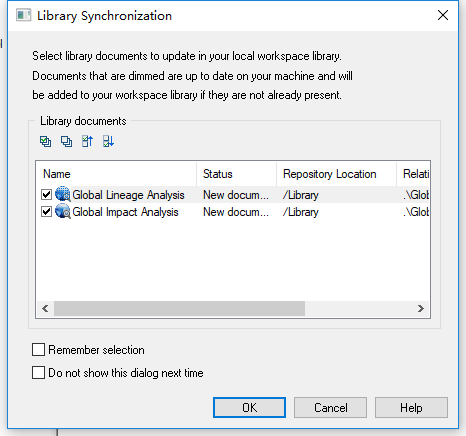




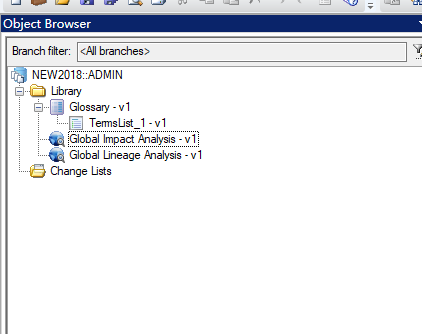
### ****创建用户****



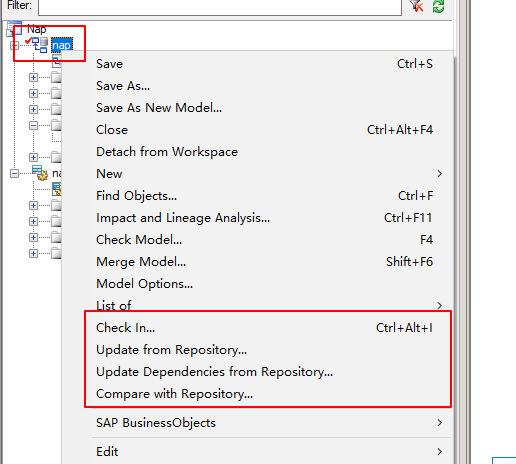
HNwhq2008



创建成功后



配置成功后的变化，

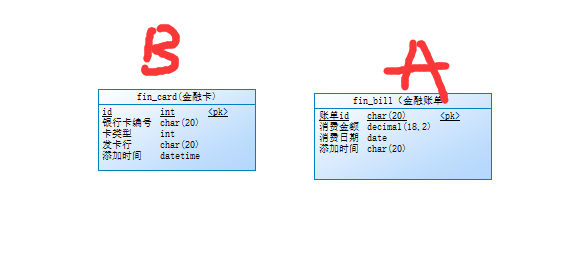


### 将设计的数据库物理模型保存保存到服务器上

### **查看模型库**

### 拉外键

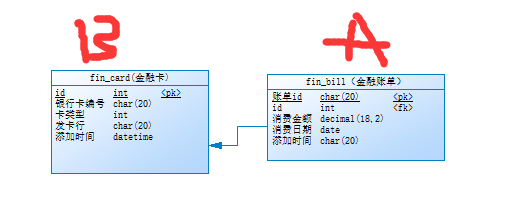
现在有A、B两个表。B中的主键是A中的外键，画图时，先建好表，然后从



使用



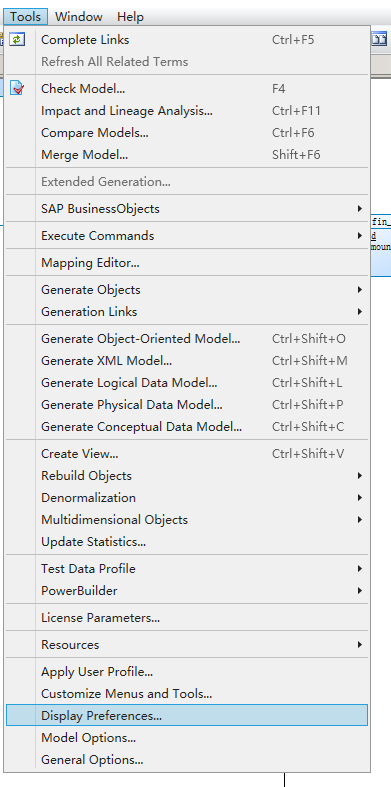
从A拉向B，则A中会增加对B的引用。

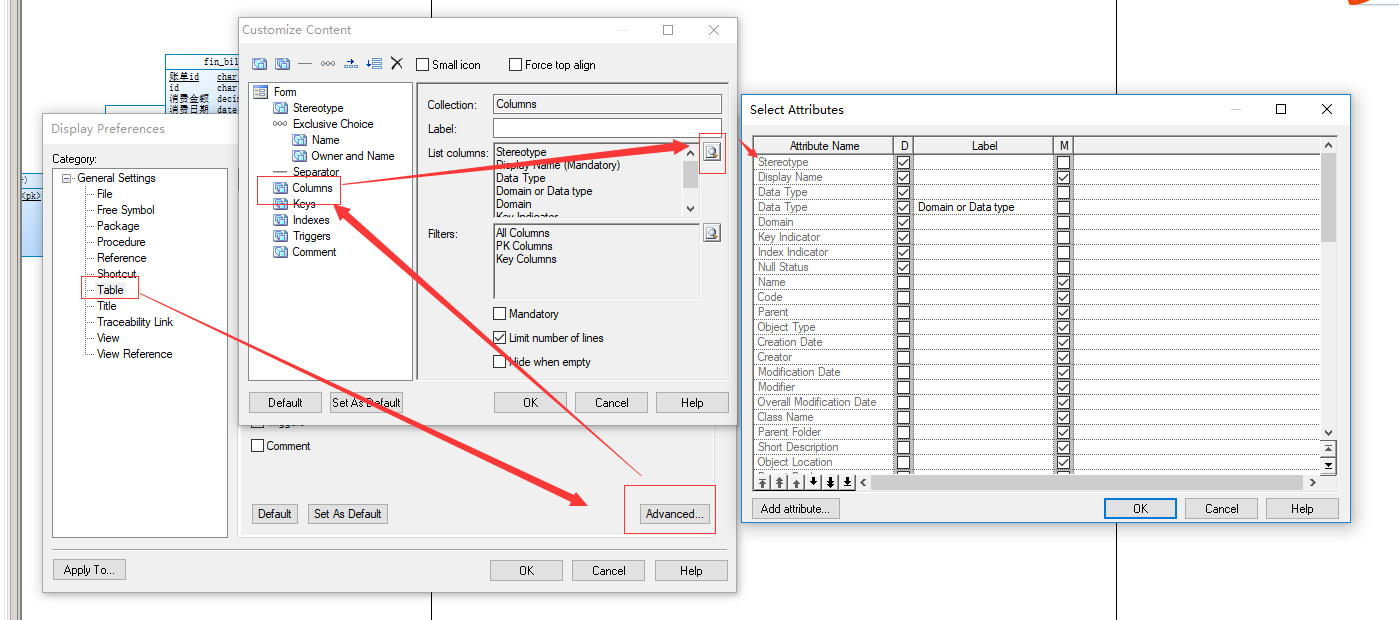


注意：如果B中主键ID的类型发生变化，引用该表中的ID字段在A表中没有发生变化。

简语：B中id 类型变化，A中外键id 没有变化。

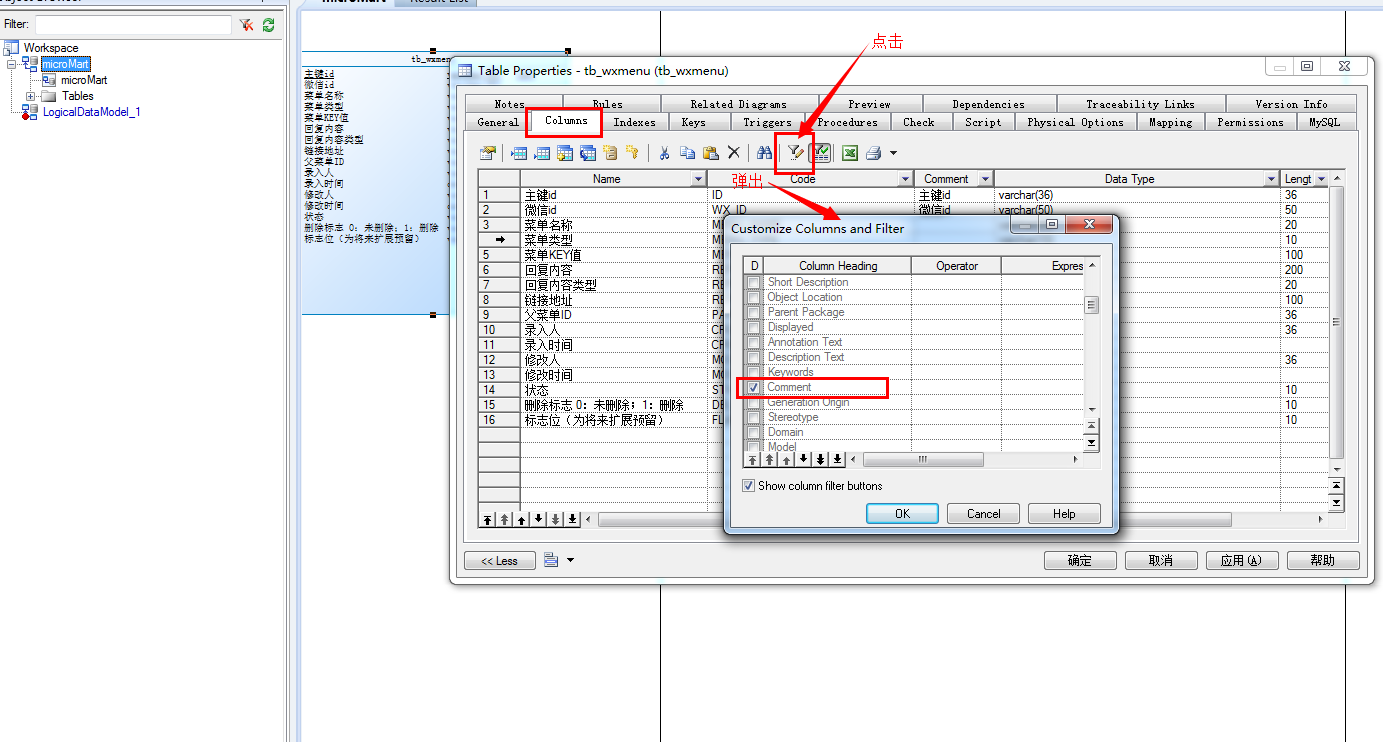
### 图形列表显示name和code等需要的信息



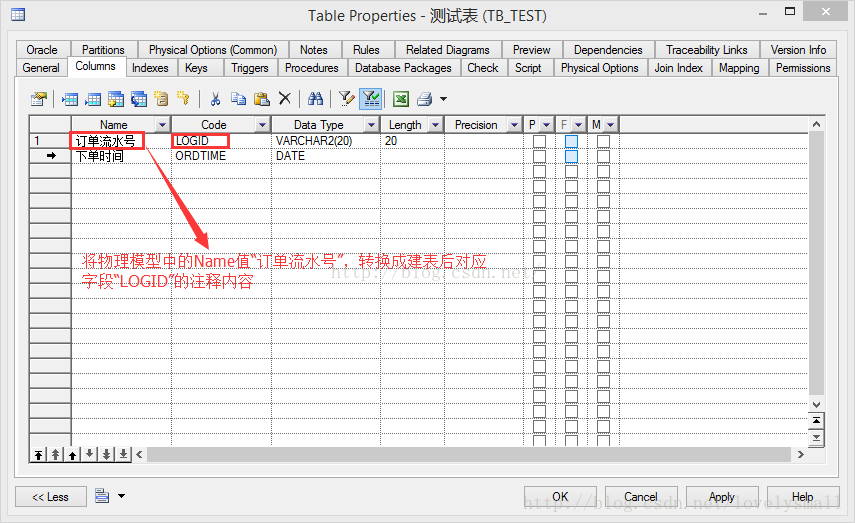
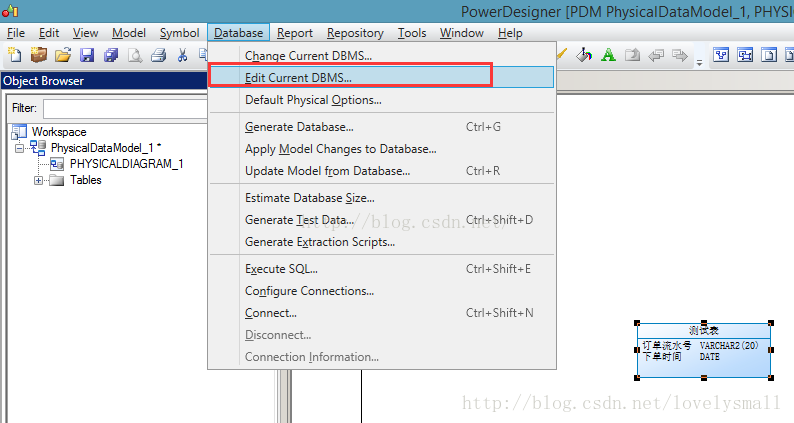
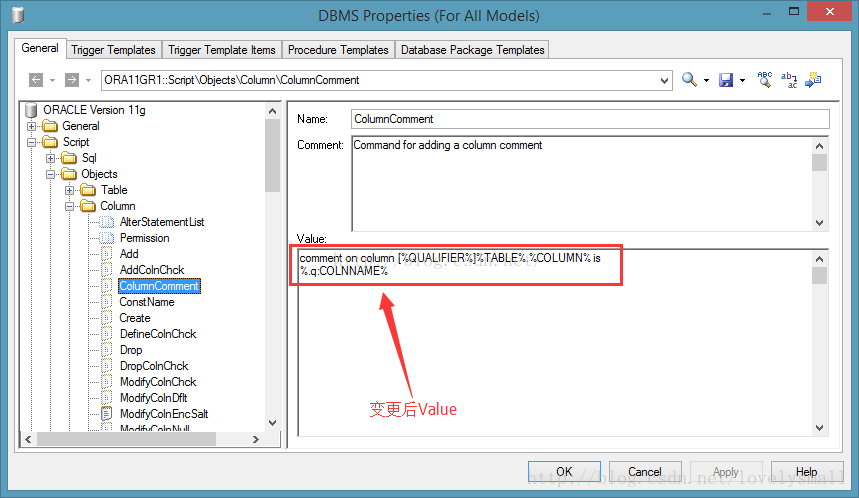
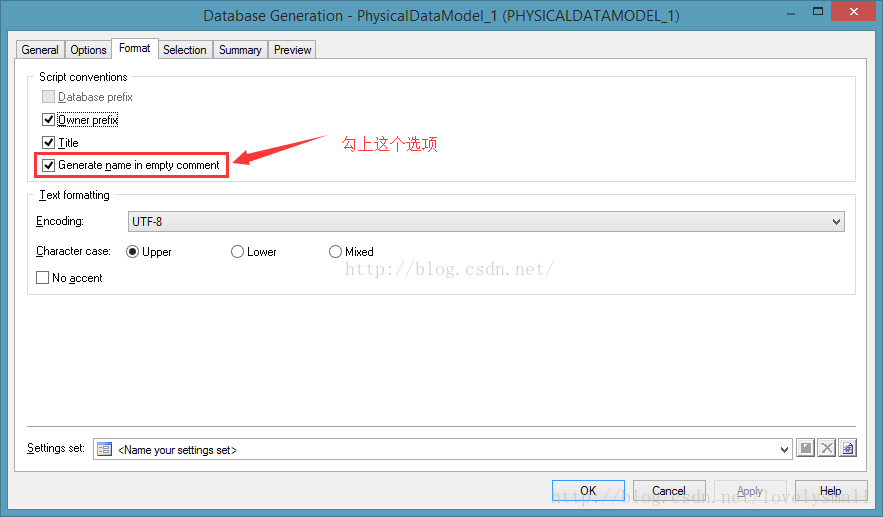
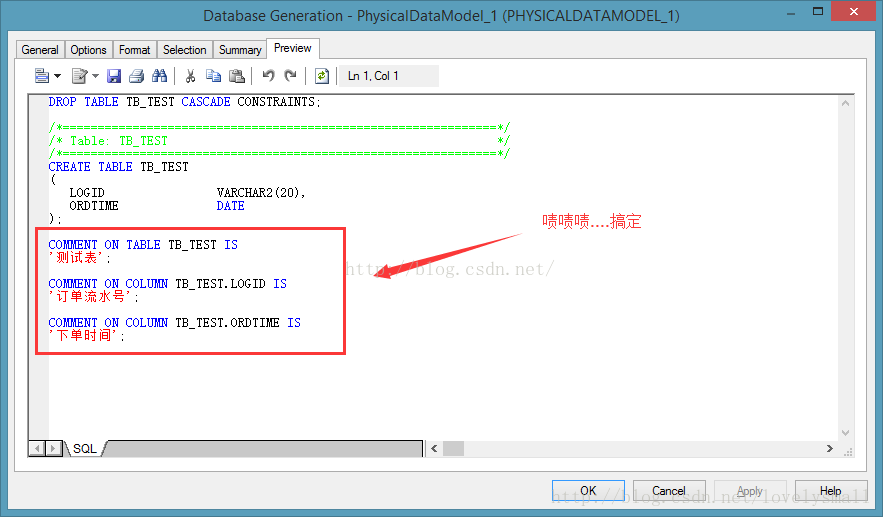


### powerdesigner 设置字段显示comment注释

在Columns标签下,一排按钮中找到这个按钮:Customize Columns and Filter

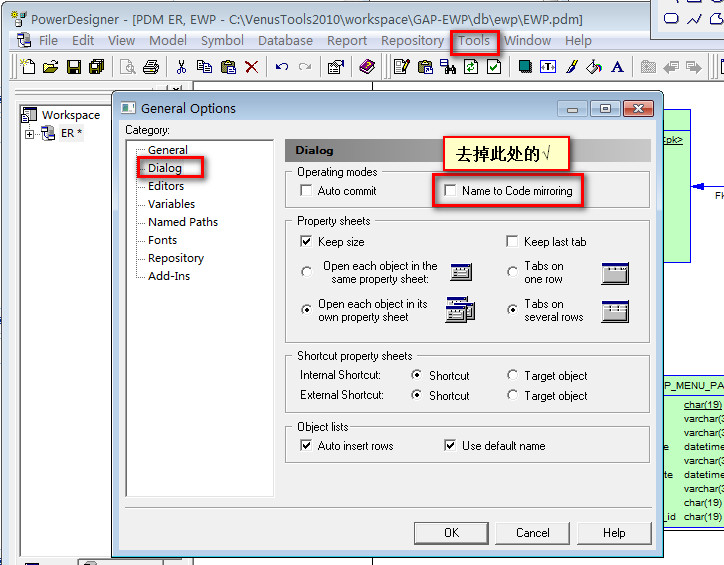


### powerdesigner 生成sql时生成对应的字段注释

本章节描述在数据库建模时，如何将物理模型中表的name值在生成数据库对应表时，转换成对应字段的注释。  
1.在物理模型中创建一个表，如下图所示：  
  
  
2.设置步骤（按图操作）  
步骤一：Database->Edit Current DBMS...  
  
  
步骤二：  
选择Edit Current DBMS...后，选择Script->Objects->Column->ColumnComment，将ColumnComment的原始内容：  
[%QUALIFIER%]%TABLE%.%COLUMN% is  
%.q:COMMENT%  
变更为  
comment on column [%QUALIFIER%]%TABLE%.%COLUMN% is  
%.q:COLNNAME%  
  
变更成功后点击【OK】，操作到此name的值就可以生成到对应表字段的注释了。  
细心的朋友可能会发现，当comment为空时，则不能生产注释，如何解决这个问题呢？请看步骤三的操作。  
  
步骤三：（解决当comment为空时，则不能生产注释的问题）  
从物理模型生产数据库时，Database->Generate Database..  
在Format标签选项中，将Generate name  in empty comment,勾选上，默认不勾选。  
  
  
可以在Preview标签选项中查看创建数据库时的SQL语句：  
  
  
到这里所有工作完成！

### 修改表的字段Name的时候Code不自动跟着变

tools-> General   Options-> Dialog:Operation   Modes:   
去掉　NameToCodeMirroring　前面的√



# 数据库外键

分布式数据库中摘掉了所有的外键。业务约束通过中间层控制，避免分表中的外键 会跨库。

## 知乎观点

取决于数据库的用途、规模、架构，有外键，可以提高鲁棒性、健壮性，但是约束检验显然会拖慢速度。

规模上说，数据量大的不适合用外键，小的可以用；

用途上安全性、可靠性很重要的就要用外键，否则可以不用。具体情况具体解决了，因为也有矛盾的时候，数据量极大，但是又要求高可靠，例如银行金融、芯片生产等，仍然需要外键的存在。可以通过SAN+RAID等硬件提升解决矛盾。要求高并发的情况下，并不适合外键，有的连关系数据库都不用了，甚至数据库都不用了。

作者：孙文亮

链接：https://www.zhihu.com/question/19600081/answer/12338663

来源：知乎

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

## 两派争论

学院派告诉你建表的时候必须使用键，而实践派除了主键和非空约束外，其他的都不要。主要通过程序来控制外键关联！

# 范式与反范式

## 范式

### 关于范式 Normal Form

范式是关系数据库理论的基础，也是我们在设计数据库结构过程中所要遵循的规则和指导方法。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，还又称完美范式）。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。**各种范式呈递次规范，越高的范式数据库冗余越小。**通常所用到的只是前三个范式，即：第一范式（1NF），第二范式（2NF），第三范式（3NF）。

* **第一范式（1NF）：**强调的是列的原子性，即列不能够再分成其他几列。简而言之，第一范式就是无重复的列。
* **第二范式（2NF）：**首先要满足它是1NF，另外还需要包含两部分内容：一是表必须有一个主键；二是没有包含在主键中的列必须完全依赖于主键，而不能只依赖于主键的一部分。

即**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**。

* **第三范式（3NF）：**在1NF基础上，**任何非主属性不依赖于其它非主属性[在2NF基础上消除传递依赖]。**第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。

简而言之，第三范式（3NF）要求一个关系中不包含已在其它关系已包含的非主关键字信息。例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性，也就是在满足2NF的基础上，任何非主属性不得传递依赖于主属性。

### 关于范式的讨论

**第二范式和第三范式如何区别？**

第二范式：非主键列是否依赖主键（包括一列通过某一列间接依赖主键），要是有依赖关系的就是第二范式；

第三范式：非主键列是否是直接依赖主键，不能是那种通过传递关系的依赖的。要是符合这种就是第三范式；

**使用范式有哪些优点和缺点？**

**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。**

范式再给我们带来的上面的好处时，同时也伴随着一些不好的地方：**按照范式的规范设计出来的表，等级越高的范式设计出来的表越多**。

如第一范式可能设计出来的表可能只有一张表而已，再按照第二范式去设计这张表时就可能出来两张或更多张表，如果再按第三范式或更高的范式去设计这张表会出现更多比第二范式多的表。

表的数量越多，当我们去查询一些数据，**必然要去多表中去查询数据，这样查询的时间要比在一张表中查询中所用的时间要高很多。也就是说我们所用的范式越高，对数据操作的性能越低。**

所以我们在利用范式设计表的时候，要根据具体的需求再去权衡是否使用更高范式去设计表。在一般的项目中，我们用的最多也就是第三范式，第三范式也就可以满足我们的项目需求，性能好而且方便管理数据；

**当我们的业务所涉及的表非常多，经常会有多表发生关系，并且我们对表的操作要时间上要尽量的快，这时可以考虑我们使用“反范式”。**

## 反范式

**不满足范式的模型，就是反范式模型。**

反范式跟范式所要求的正好相反，在反范式的设计模式，我们可以允许适当的数据的冗余，用这个冗余去取操作数据时间的缩短。**本质上就是用空间来换取时间，把数据冗余在多个表中，当查询时可以减少或者是避免表之间的关联；**

RDBMS模型设计过程中，常常使用范式约束我们的模型，但在NOSQL模型中则大量采用反范式。

### 范式和反范式的对比

|  |  |  |
| --- | --- | --- |
| 范式化模型 | 数据没有冗余，更新容易 | 当表的数量比较多，  查询设计需要很多关联模型（join）时，会导致查询性能低下 |
| 反范式化模型 | 数据冗余将带来很好的读取性能  （因为不需要join很多表，而且通常反范式模型很少做更新操作） | 需要维护冗余数据，从目前NoSQL的发展可以看到，  对磁盘空间的消耗是可以接受的 |

# VM

