文件修改控制页

修改记录编号	修改内容	修改人	修改日期

[附加说明]

1. “修改记录编号”的填写内容为:本次修改后的版本号+“/”+流水号,例如:V1.01/1。
一次修改可以修改文档的多个位置,流水号为对该版本修改的流水号。当版本变时,流水号归为1。

目 录

1 设计文件内容

数据库设计完成后，应形成并提交以下的设计文件：

□1□ 数据库脚本文件

是用于创建数据库、生成数据库表、视图等内容的脚本文件，文件后缀为.sql，包括：

- create_database.sql　　创建数据库
- create_user.sql　　创建数据库用户
- create_table.sql　　创建数据库表、视图、索引
- create_data.sql　　创建系统初始化数据

（2）原始设计文件

是指使用Power Designer进行设计的文件，如：.pdm文件。

（3）其它必要文件

2 命名规范

2.1 设计文件的命名

数据库脚本文件应保持与上述的命名一致。

2.2 数据库文件的命名

Oracle数据库的文件类型有4种：数据文件、联机重做日志文件、控制文件、参数文件，每种文件的命名应遵循如下的约定。

2.2.1 数据文件的命名

数据文件与表空间具有多对一的关系，即，一个表空间可以有一个或多个数据文件，一个数据文件只能属于一个表空间，因此可以通过在文件名中增加表空间名来区分数据文件的归属，格式为：

<表空间名><2位数字>.dbf

例如：empty01.dbf、empty02.dbf

2.2.2 联机重做日志文件的命名

联机重做日志文件记录与数据库有关的重做数据，联机重做日志组成重做日志组，每个重做日志组可由多个成员组成，每组必须由相同个数的成员组成。命名格式为：

redo<member_number><group_identifier>.log

member_number为2位数字，表示日志组中日志文件的序号；

group_identifier为单一字符，表示重做日志组；

例如：组A　redo01a.log　redo02a.log

组B　redo01b.log　redo02b.log

2.2.3 控制文件的命名

控制文件存储数据库的关键性结构信息，每个Oracle数据库应该至少有两个（最好三个）控制文件，命名约定如下：

control<nn>.ctl

2.2.4数据库参数文件的命名

在Oracle安装完毕后，系统在$ORACLE_HOME/dbs目录下自动生成一个参数初始化文件。该文件的命名约定如下：

init<SID>.ora　　SID是Oracle唯一的实例标识符

另外，用户还可以建立其他的参数文件，这些文件可以在主要的参数配置文件中通过IFILE参数来指定。利用IFILE功能，能够更好的组织参数文件以及在多个实例中共享某些配置。其命名约定如下：

config<SID>.ora

2.3表命名

□1□ 使用有意义的英文单词来命名
□2□ 表名用大写英文字母表示
□3□ 以字母开头，其余部分可以是字母、数字的任意组合
□4□ 使用“_”来分割单词，如ACCOUNT_HISTORY
□5□ 表名的长度限制在30个字符之内
□6□ 表名以一个模块名的缩写作为前缀，如SYS_USER
□7□ 子表名与父表名有相应的联系，如：父表ACCOUNT、子表ACCOUNT_HISTORY
□8□ 表名不得是Oracle数据库的保留字

2.4视图命名

规则：“VIEW_表A_表B”

其中“VIEW_”作为前缀，以区别于表名，“表A、表B”表示组成视图的表名，如果表过多或其中若干个表名过长，则可以适当简化视图名

例如：VIEW_TICKET是从表TICKET、TICKET_SOURCE、TICKET_PRICE_POLICY三个表中提取数据而形成的一个关于票的完整信息。

2.5索引命名

规则：ix_<表名>_n

例如：ix_account、ix_account_1、ix_account_2

2.6 Constraint命名

Check Constraint规则：chk_<表名>_<列名>

2.7字段命名

□1□ 使用有意义的英文单词来命名
□2□ 字段名用大写英文字母表示
□3□ 使用“_”来分割单词
□4□ 字段名的长度限制在30个字符之内
□5□ 使用后缀来表示列的数据类型或特定含义，具体内容见下表

后缀	数据类型	描述	例子

*_id	Number(10)	通常用来做主键名称的后缀，这种主键没有实际意义，仅仅给出每一条记录的一个标识，该字段通常基于一个SEQUENCE。	sms_id
*_no	Varchar2(10)	用来做主键名称的后缀，这种主键有实际意义	staff_no
*_dt	Date/Datetime	时间和日期类型	create_dt
*_code	Varchar2(10)	编码类型，表示该字段存储的值是一种预定义的编码。这个编码一定存在于MAS_CODE表中，含有*_code字段的表和MAS_CODE表建立主外键关系	status_code、error_code
*_flag	Varchar2(1)	用来表示“Yes” or “No”，其值用大写的“Y”、“N”表示	start_flag
*_price	Number(10,2)	金额或货币的表示	ticket_price
*_name	Varchar2(80)	表示对象名称	user_name
*_age	Number(3)	表示年龄	user_age
*_remark	Varchar2(255)	注释、备注等	
*_desc	Varchar2(255)	对象的描述信息	user_desc

2.8 SEQUENCE的命名

规则：“SEQ_序列名”，其中以“SEQ_”开头，序列名用有意义的英文单词表示，是字母、数字和“_”的任意组合。

例如：SEQ_USER_ID

2.9 Key命名

主键（Primary Key）：Pk_<表名>　　例如：pk_account

外键（Foreign Key）：fk_<表名>_<列名>　例如：fk_account_account_type

3设计原则

□1□ 所有记录业务数据的表中都要设有“操作员”、“记录创建日期”、“记录修改日期”的字段。

□2□ varchar2类型的字段长度尽量使用10、30、80、255、2000中的一种。但是，含有特定意义的字段如IP地址等，可以使用习惯的长度。

□3□ 子表的主键选择：对于第二层的子表，主键通常由父表的主键加上子表自身的主键共同组成；对于第三层以上的子表，只使用子表自身的主键，而父表的主键作为外键存在。

□4□ 具有master_detail关系的表的字段顺序，假设account和account_history是两个具有master_detail关系的表，通常在account表中account_id是主键，是account_history表的外键。在这种情况下，要

求account_history表的第一个字段为account_id，第二个字段为account_history_id。

□5□ code表用来存储预定义的一些编码信息，所有预定义的数据信息都要在数据库中作为系统初始化数据录入，在需要的时候可以作维护界面对其进行管理。但是code表中的记录数不要超过4000条，以免影响系统性能。Code表的结构

列名	数据类型	描述	主键
master_type	Varchar2(10)	Code的种类	Yes
code_type	Varchar2(10)	Code的值	Yes
code_desc	Varchar(255)	Code的描述	

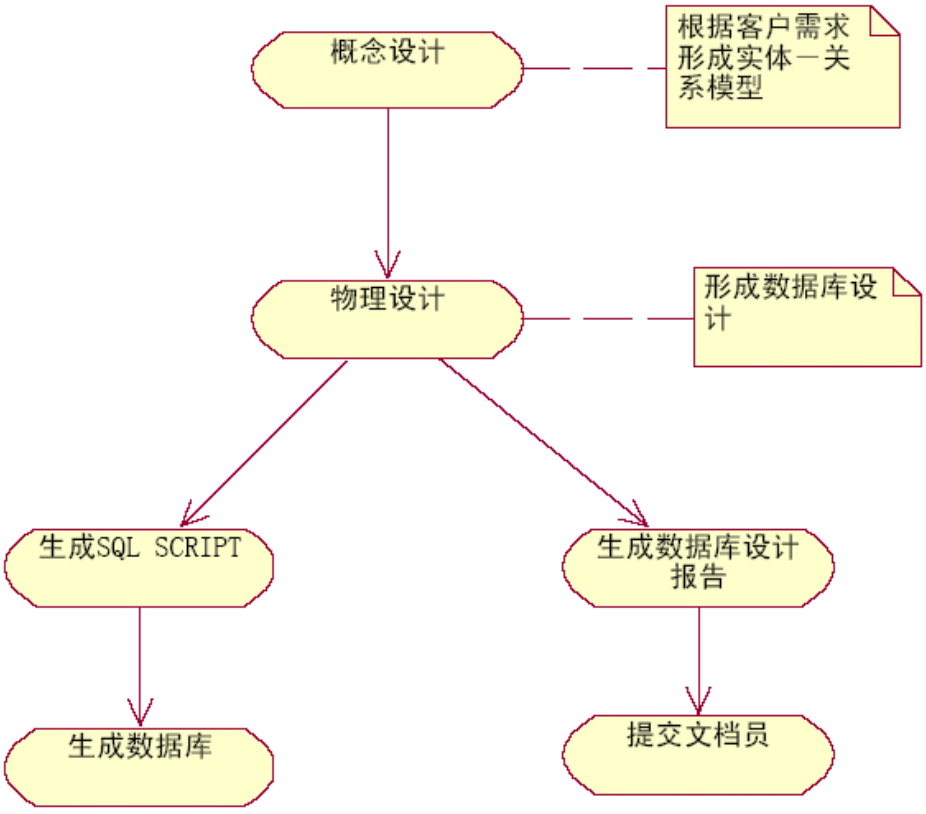
□6□ 数据库脚本文件的编写要求

- 所有的SQL保留字必须大写；
- INSERT语句必须指定列名；
- WHERE子句中的列应该给出表的别名；

4附录：Power Designer使用介绍

在数据库设计过程中，使用Power Designer作为统一的数据库设计辅助工具，数据库结构的设计和创建数据库的脚本都使用Power Designer。

设计数据库的过程大致如下：



4.1概念设计

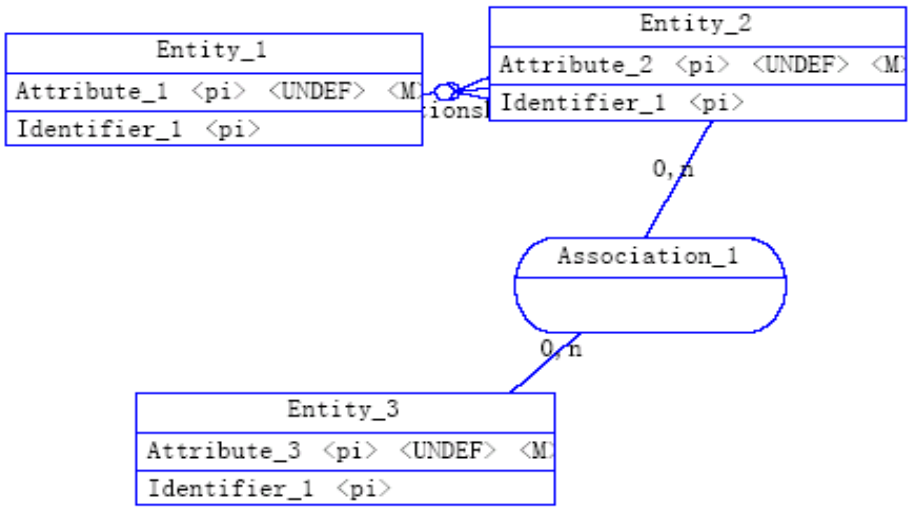
所谓概念设计是根据用户的需求建立实体－关系模型，搞清楚应用系统要处理的数据以及它们之间的关系。概念设计最终产生的结果是一份实体－关系图。

在Power Designer中选择File－>New，在“新建”对话框中选择“Conceptual Data Model”。在Workspace下会出现“ConceptualDataModel_1”，展开后可看到“Diagram_1”，双击Diagram_1可以在右边的设计窗口进行编辑。在Conceptual Data Model中有Entity、Relationship、Association三个基本元素：

Entity为实体，表示应用系统中需要处理的一个个数据对象；

Relationship可以连接两个实体，标识他们之间的关系，两个实体间的关系可以是一对一、一对多和多对多。

在Power Designer中双击Relationship，可以在其属性框的detail选项卡中选择这些关系。当两个Entity之间存在多对多的关系时，通常会在这两个Entity之间建立一个Association，如下图：

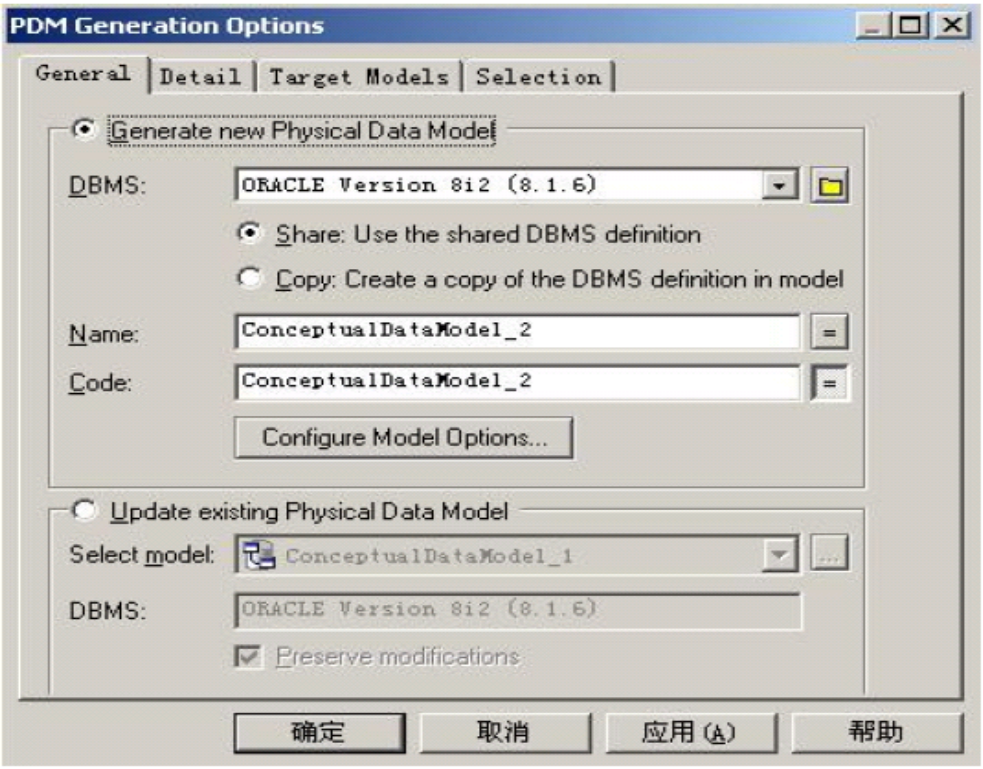


概念设计是从较高的层次对应用系统所需要处理的数据进行组织和图形化，它不涉及具体的数据库管理系统。Power Designer创建的Conceptual Data Model保存后生成一个*.cmd的文件。

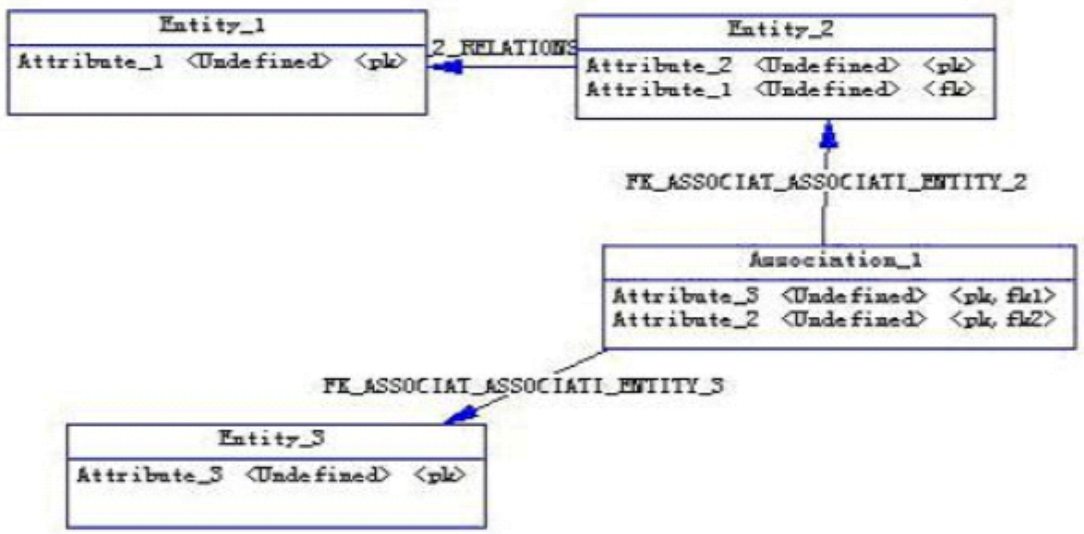
4.2物理设计

确定实体－关系图后就可以开始物理设计。所谓物理设计是对如何将概念设计部署到某一具体的数据库管理系统的设计。

在Power Designer中通过已经确定的Conceptual Data Model自动生成一个Physical Data Model。选择Tools－>Generate Physical Data Model，可以看到一个“PDM Generation Option”对话框



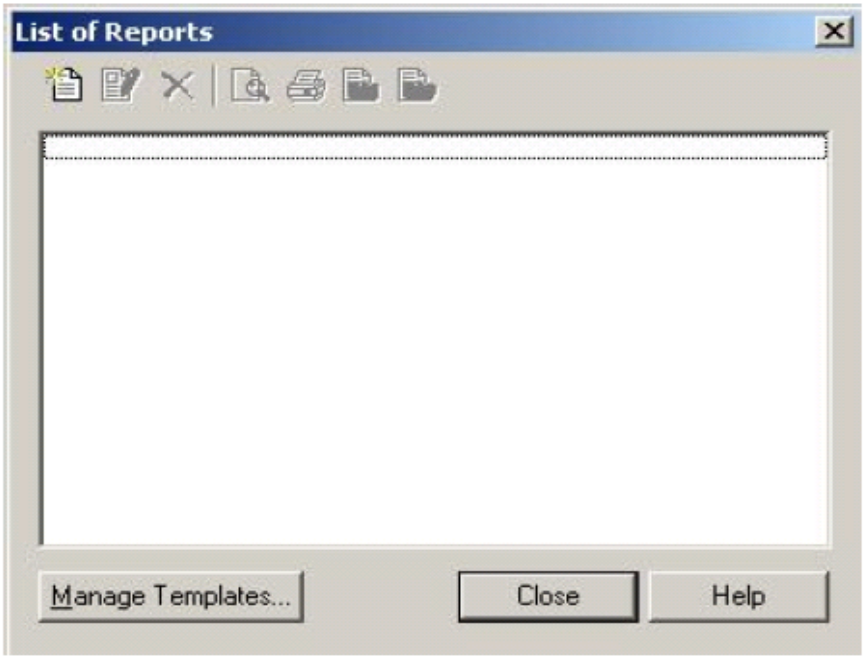
选择一个数据库管理系统(Oracle 8.1.6)后，单击“确定”，Power Designer自动根据Conceptual Data Model产生Physical Data Model，如下图所示。



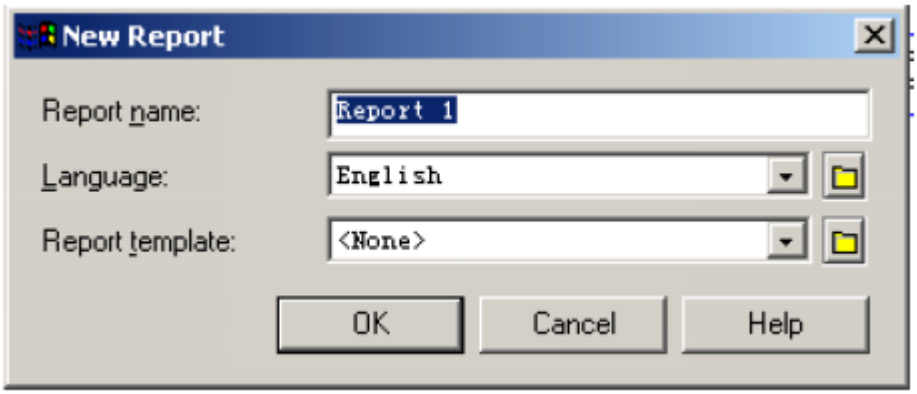
在这张图中，Entity和Association分别对应数据库中的一张表，而Relationship会被映射成一个主外键约束。同时，数据库表的每个字段都没有明确的数据类型。

4.3产生数据库设计报告(数据字典)

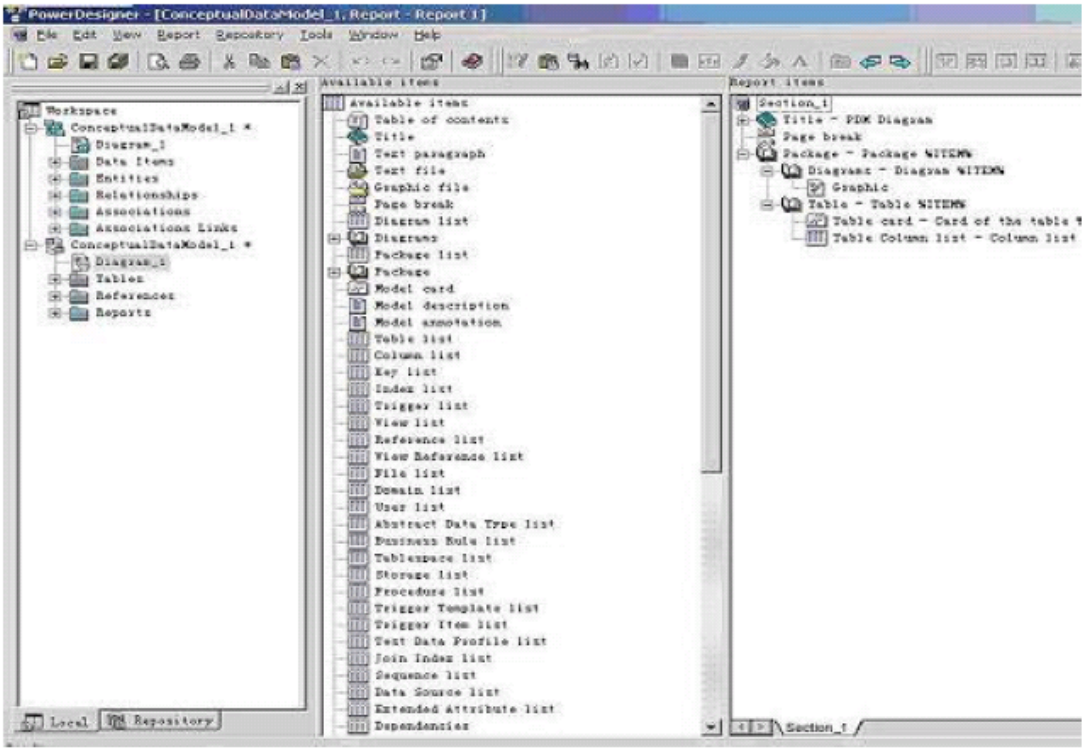
完成物理设计后，为了便于维护和交流需要生成数据库设计报告。Power Designer提供报表生成的模板，我们已经建立了生成报表的模板文件(DB_Report_Pkg_Template.rtp，附于此文档同目录下)选择Model->Report，会看到如下的界面。



点击左上角的“New Report”按钮，会看到“New Report”对话框：



在对话框中输入数据库设计报告的名字，采用的语言，通过点击“Report template”右边的小按钮选择模板所在的目录，再下拉Report template下拉框选择“Design Report”，点击“OK”按钮，可以看到报告定制界面



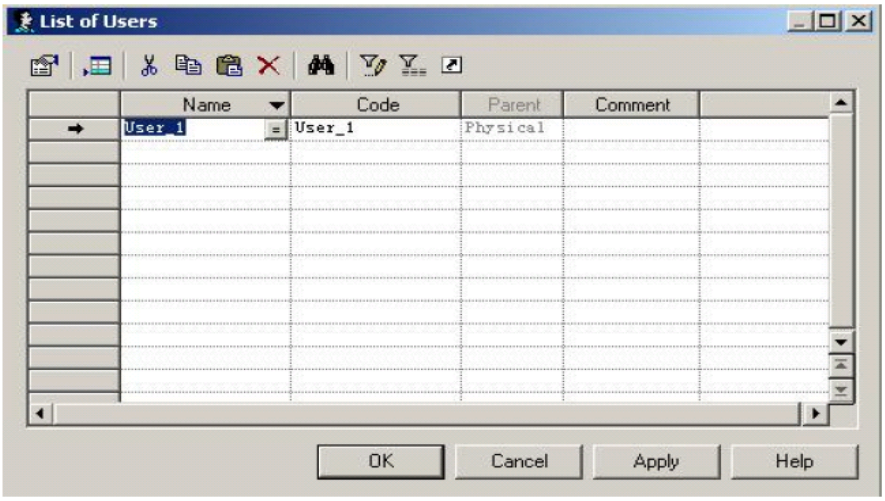
现在，可以通过选择File—>Generate—>RTF菜单生成数据库设计报告了。

4.4生成数据库脚本

在设计完成后，需要利用Power Designer生成数据库脚本(创建数据库结构、初始化数据的SQL语句)，利用这些数据库脚本可以方便的在具体的数据库管理系统上部署项目的数据库。

4.4.1生成create_user.sql

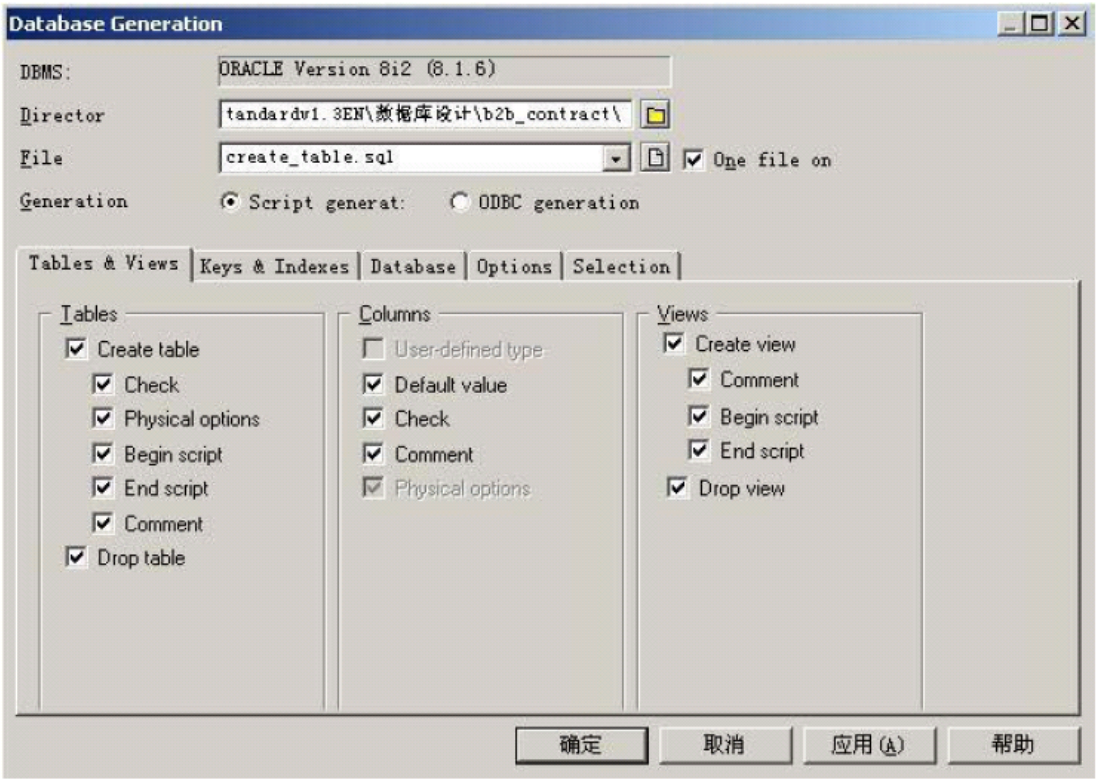
点击Model—>User...，可以看到如下的对话框：



点击左上角的“属性”图标可以编辑该用户的属性。Power Designer不能自动生成create_user.sql，该文件需要手工编写。

4.4.2生成create_table.sql

选择Database一>Generate Database…，可以看到Database Generate对话框。



Power Designer生成的SQL Script是一个*.sql的文件，在这个对话框中可以选择文件存放的路径以及文件中需要包含的各种信息，如Table中的Check、Physical Options等，通常情况下不需要修改这些默认。