

# 电子科技大学

## 实验指导书

课程名称：数据库应用基础



# 目 录

<b>第一章 创建、备份与恢复数据库.....</b>	<b>1</b>
1.1 实验原理.....	1
1.2 实验目的.....	1
1.3 实验内容与要求.....	1
1.4 实验器材（设备、元器件） .....	2
1.5 实验指导.....	2
<b>第二章 数据库的完整性.....</b>	<b>7</b>
2.1 实验原理.....	7
2.2 实验目的.....	7
2.3 实验内容与要求.....	7
2.4 实验器材（设备、元器件） .....	8
2.5 实验指导.....	8
<b>第三章 数据的修改.....</b>	<b>11</b>
3.1 实验原理.....	11
3.2 实验目的.....	11
3.3 实验内容与要求.....	11
3.4 实验器材（设备、元器件） .....	11
3.5 实验指导.....	11
<b>第四章 简单查询、多表查询.....</b>	<b>15</b>
4.1 实验原理.....	15
4.2 实验目的.....	15
4.3 实验内容与要求.....	15
4.4 实验器材（设备、元器件） .....	15
4.5 实验指导.....	16

<b>第五章 分组统计查询.....</b>	<b>17</b>
5.1 实验原理.....	17
5.2 实验目的.....	17
5.3 实验内容与要求.....	17
5.4 实验器材（设备、元器件） .....	18
5.5 实验指导.....	18
<b>第六章 集合操作、子查询.....</b>	<b>22</b>
6.1 实验原理.....	22
6.2 实验目的.....	22
6.3 实验内容与要求.....	22
6.4 实验器材（设备、元器件） .....	22
6.5 实验指导.....	22
<b>第七章 数据库建模.....</b>	<b>31</b>
7.1 实验原理.....	31
7.2 实验目的.....	31
7.3 实验内容与要求.....	31
7.4 实验设备与环境.....	31
7.5 实验指导.....	31
<b>第八章 POWERDESINGER 操作指南.....</b>	<b>38</b>
8.1 POWERDESIGNER 简介.....	38
8.2 使用 POWERDESIGNER 环境.....	40
8.3 将 CDM 对象转换成 PDM 对象.....	57
8.4 正向工程和逆向工程.....	61
<b>第九章 MS SQLSERVER 2008 简介.....</b>	<b>68</b>
9.1 操作简介.....	68
9.2 MS SQLSERVER 函数.....	73

---

# 第一章 创建、备份与恢复数据库

## 1.1 实验原理

使用数据库管理系统 DB、DDL 创建数据库及数据库对象。

## 1.2 实验目的

本实验要求学生掌握创建数据库的方法及相关操作，创建数据库，向数据库中添加样本数据，学习 SQLSERVER 数据库的恢复和备份。

## 1.3 实验内容与要求

- (1) 启动 SQLSERVER
- (2) 创建数据库：STUD
- (3) 创建表
- (4) 插入样本数据
- (5) 备份数据库
- (6) 恢复数据库

“系别代码表 “	表名: dep
“教师表”	表名: teacher
“学生表”	表名: student
“课程表”	表名: course
“选课表”	表名: sc

为每个表准备大约 10 记录，使用 Insert 语句将这些数据插入到相应表中  
数据录入完成后，将数据库备份到磁盘上，在以后的的试验中备用。

---

## 1.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP

数据库：MS SQLSERVER

## 1.5 实验指导

### 1.5.1 结构化查询语言

SQL 已经确立起自己作为标准关系数据库语言的地位。SQL 有许多种版本，最早的版本是由 IBM 的 San Jose 研究室提出的。该语言最初叫做 SQUEL (Structured English Query Language)，作为 System R 项目的一部分于 70 年代初付诸实施，发展到现在，它的名字已变为 SQL（结构化查询语言）。

1986 年，美国国家标准协会（ANSI）和国际标准化组织（ISO）发布了 SQL 标准 SQL-86。在 1989 年，这一标准升级为 SQL-89。现在的数据库系统至少要支持 SQL-89。ANSI/ISO 的最新标准是 SQL-92。SQL 标准的下一版本现正在制定中，暂时叫做 SQL-3。

SQL 语言可分为以下三部分：

- 数据定义语言（DDL）。使用 DDL 可创建和删除表、模式、域、索引和视图，并可对它们进行修改。
- 数据操纵语言（DML）。不仅包括基于关系代数和元组关系演算的查询语言，还包括在数据库中插入、删除、修改元组的命令。
- 数据控制语言（DCL）。数据控制包括安全性控制、完整性控制、事务控制和并发控制。

#### 1.5.1.1 数据定义语言（DDL）（data definition language）

数据定义语言是指对资料的格式和形态下定义的语言，他是每个资料库要建立时候时首先要面对的，举凡资料分哪些表格关系、表格内的有什么栏位主键、表格和表格之间互相参考的关系等等，都是在开始的时候所必须规划好的。

1、建表：

```
create table table_name(  
column1 datatype [not null] [not null primary key],  
column2 datatype [not null],  
...)
```

---

说明:

`datatype` --是字段的数据类型, 详见表。

`not null` --可不可以允许字段有空值 (尚未有资料填入)。

`primary key` --是本表的主键。

## 2、更改表

```
alter table table_name
```

```
add column column_name datatype
```

说明: 增加一个字段

```
alter table table_name
```

```
add primary key (column_name)
```

说明: 更改表的定义把某个字段设为主键。

```
alter table table_name
```

```
drop primary key (column_name)
```

说明: 把主键的定义删除。

## 3、建立索引

```
create index index_name on table_name (column_name)
```

说明: 对某个表格的字段建立索引以增加查询时的速度。

## 4、删除

```
drop table_name
```

```
drop index_name
```

### 1.5.1.2 数据类型 datatypes

#### (一) 整数数据

- `bit`: `bit` 数据类型代表 0, 1 或 `NULL`, 就是表示 `true, false`. 占用 1byte.
- `int`: 以 4 个字节来存储正负数. 可存储范围为:  $-2^{31}$  至  $2^{31}-1$ .
- `smallint`: 以 2 个字节来存储正负数. 存储范围为:  $-2^{15}$  至  $2^{15}-1$
- `tinyint`: 是最小的整数类型, 仅用 1 字节, 范围: 0 至此 $2^8-1$

#### (二) 精确数值数据

- `numeric`: 表示的数字可以达到 38 位, 存储数据时所用的字节数目会随着使用位数多少变化.
- `decimal`: 和 `numeric` 差不多

---

### （三）近似浮点数值数据

- float:用 8 个字节来存储数据.最多可为 53 位.范围为:-1.79E+308 至 1.79E+308.
- real:位数为 24,用 4 个字节,数字范围:-3.04E+38 至 3.04E+38

### （四）日期时间数据

- datetime:表示时间范围可以表示从 1753/1/1 至 9999/12/31,时间可以表示到 3.33/1000 秒.使用 8 个字节.
- smalldatetime:表示时间范围可以表示从 1900/1/1 至 2079/12/31.使用 4 个字节.

### （五）字符串数据

- char:长度是设定的,最短为 1 字节,最长为 8000 个字节.不足的长度会用空白补上.
- varchar:长度也是设定的,最短为 1 字节,最长为 8000 个字节,尾部的空白会去掉.
- text:长宽也是设定的,最长可以存放 2G 的数据.

### （六）Unicode 字符串数据

- nchar:长度是设定的,最短为 1 字节,最长为 4000 个字节.不足的长度会用空白补上.储存一个字符需要 2 个字节.
- nvarchar:长度是设定的,最短为 1 字节,最长为 4000 个字节.尾部的空白会去掉.储存一个字符需要 2 个字节.
- ntext:长度是设定的,最短为 1 字节,最长为 2G.尾部的空白会去掉,储存一个字符需要 2 个字节.

### （七）货币数据类型

- money:记录金额范围为:-92233720368577.5808 至 92233720368577.5807.需要 8 个字节.
- smallmoney:记录金额范围为:-214748.3648 至 214748.36487.需要 4 个字节.

### （八）标记数据

- timestamp:该数据类型在每一个表中是唯一的!当表中的一个记录更改时,



---

该记录的 timestamp 字段会自动更新。

- **uniqueidentifier**: 用于识别数据库里面许多个表的唯一一个记录。

### (九) 二进制码字符串数据

- **binary**: 固定长度的二进制码字符串字段, 最短为 1, 最长为 8000。
- **varbinary**: 与 **binary** 差异为数据尾部是 00 时, **varbinary** 会将其去掉
- **image**: 为可变长度的二进制码字符串, 最长 2G。

## 1.5.2 实验步骤

### (1) 创建数据库: STUD

```
Create database stud;
```

### (2) 创建表, 表结构见下图, 表的详细说明见图后的表格.

#### “系别代码表” 表名: dep

```
create table DEP(DEPID CHAR(8) not null, DEPNAME CHAR(20) not null, primary key (DEPID));
```

#### “教师表” 表名: teacher

```
create table TEACHER(TID varchar(8) not null, TNAME varchar(8) not null, TITLE VARCHAR(8) not null, DEPID VARCHAR(8), primary key (TID));
```

#### “学生表” 表名: student

```
create table STUD(SID VARCHAR(11) not null, DEPID CHAR(8) not null, SNAME VARCHAR(8) not null, sex char(2) not null, BIRTHD DATE not null, EMAIL VARCHAR(40), HOMEADDR VARCHAR(40), primary key (SID));
```

#### “课程表” 表名: course

```
create table COURSE(CID VARCHAR(8) not null, CNAME VARCHAR(20) not null, CID_PRE VARCHAR(8), CREDITS NUMERIC(3,1) not null, primary key (CID));
```

#### “选课表” 表名: sc

```
create table SC(SID VARCHAR(11) not null, CID VARCHAR(8) not null, TID varchar(8) not null, SCORE INTEGER, primary key (sid, cid, tid));
```

### (3) 备份数据库

```
BACKUP DATABASE stud TO DISK = 'd:\stud.bak' WITH INIT;
```

### (4) 恢复数据库

---

```
RESTORE DATABASE stud FROM DISK = 'd:\stud.bak' WITH REPLACE;
```

---

## 第二章 数据库的完整性

### 2.1 实验原理

数据库的完整性、约束条件、结构化查询语言。

### 2.2 实验目的

通过设置表的检查约束、外键约束体会数据库完整性的含义，约束条件下数据修改操作的限制，以及实现修改操作的技巧。

### 2.3 实验内容与要求

#### (1) 添加约束条件

- 设置选课表的三个外键约束（学号，课程号，教师号）
- 设置教师表，学生表中的院系字段(dep\_id)的外键约束
- 设置学生表中姓名字段为 非空字段 （必须有数据，不能是空值）
- 设置选课表中成绩字段的取值范围是 0 到 100
- 设置学生表中性别字段的取值为 “男” 或 “女”
- 设置学生表电子邮件字段的取值必须包含@符号

#### (2) 添加样本数据

- 向 “系别代码表 “ 添加数据：

```
insert into dep values ('601','计算机科学与工程');
insert into dep values ('603','信息安全');
```

- 向 “教师表” 添加数据

```
insert into teacher values ('T01','教师 1','教授','601');
insert into teacher values ('T02','教师 2','工程师','601');
```

- 向 “学生表” 添加数据，至少 10 条以上

```
insert into stud(SID,DEPID,SNAME,sex,BIRTHD,EMAIL,HOMEADDR) values (...);
```

- 向 “课程表” 添加数据

```
insert into course values ('6001','计算机组成原理',null,3);
```

---

```
insert into course values ('6002','操作系统','6001',3);
insert into course values ('6003','数据结构',null,3);
insert into course values ('6004','数据库原理',null,3);
```

- 向“选课表” 添加数据

```
insert into sc(sid,cid,tid) values (...);
```

(3) 将学号为“601”学生的学号改为“20060601”，且同时更改该所有的选课信息。

## 2.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP；数据库：MS SQLSERVER

## 2.5 实验指导

### 2.5.1 主键 Primary Key

```
Alter table <table_name>
    add constraint <constraint_name> Primary key(<column_name1>)
```

例如：

```
Alter table student
    Add constraint pk_student Primary key (student_id);
```

### 2.5.2 外键 Foreign Key

```
Alter table <table_name>
    add constraint <constraint_name> Foreign key(<column_name>)
    References <table_name2> (<column_name>)
```

例如：

```
Alter table student
```

---

```
Add constraint fk_stud_dep Foreign key (depid)
References department (depid);
```

### 2.5.3 数据有效性检查 Check

```
Alter table <table_name>
Add constraint <constraint_name> check (<表达式>);
例如:
Alter table student
add constraint ck_stud_sex check (sex in ('男','女'));
```

### 2.5.4 实验步骤

1. 恢复数据库，将数据库备份 stud 恢复到 SQLSERVER 中

```
Restore db stud from d:\
```

2. 添加约束条件  
建立外键

```
alter table SC add constraint F_Reference_3 foreign key (SID) references STUD
(SID)
```

```
alter table SC add constraint F_Reference_4 foreign key (CID) references COURSE
(CID)
```

```
alter table SC add constraint F_Reference_5 foreign key (TID) references TEACHER
(TID)
```

```
alter table STUD add constraint F_Reference_2 foreign key (DEPID) references
DEP (DEPID)
```

```
alter table TEACHER add constraint F_Reference_1 foreign key (DEPID) references
DEP (DEPID)
```

设置检查条件

```
alter table course add constraint chk_course_1 check (cid_pre<> cid and cid_pre
```

---

```
in cid)
```

```
alter table stud add constraint chk_stud_1 check (email like '_%@_%')
```

### 3. 添加样本数据

- 向“系别代码表” 添加数据:

```
insert into dep values ('601','计算机科学与工程');
```

```
insert into dep values ('603','信息安全');
```

- 向“教师表” 添加数据

```
insert into teacher values ('T01','教师1','教授','601');
```

```
insert into teacher values ('T02','教师2','工程师','601');
```

- 向“学生表” 添加数据, 至少 10 条以上

```
insert into stud(SID,DEPID,SNAME,sex,BIRTHD,EMAIL,HOMEADDR) values (...);
```

- 向“课程表” 添加数据

```
insert into course values ('6001','计算机组成原理',null,3);
```

```
insert into course values ('6002','操作系统','6001',3);
```

```
insert into course values ('6003','数据结构',null,3);
```

```
insert into course values ('6004','数据库原理',null,3);
```

- 向“选课表” 添加数据

```
insert into sc(sid,cid,tid) values (...);
```

### 4. 备份数据库

```
BACKUP DATABASE stud TO DISK = 'd:\stud.bak' WITH INIT;
```

---

## 第三章 数据的修改

### 3.1 实验原理

使用结构化查询语言，在满足约束条件的情况下完成数据修改

### 3.2 实验目的

练习 UPDATE、DELETE 命令的使用，实现对数据的修改和删除。

### 3.3 实验内容与要求

- (1) 将院系中，原院系名“IS”改为“Information ”
- (2) 在选课表中，删除计算机科学与工程系学生选修 2 号课程的记录
- (3) 在选课表中，删除软件工程系学生选课 1 号课程的纪录记录
- (4) 登记考试成绩，数据自拟：
- (5) 学号为 2406010103 的同学由原来的计算机科学与工程系转入信息安全系，学号更改为 2406030102，在数据库中做出相应修改。

### 3.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP

数据库：MS SQLSERVER

### 3.5 实验指导

#### 3.5.1 删除记录 (delete)

删除请求的表达与查询非常类似。可以删除整个元组，但不能只删除某些属性上的值。其格式如下：

```
delete from R
```

---

```
where F
```

其意义是：从关系 R 中删除满足条件 F 的元组。如果省略 F 语句，则 R 中所有元组都被删除。

应注意：

1. delete 命令只对一个关系起作用。如果想从多个关系中删除元组，就要为每个关系写一条 delete 命令。

delete 子句中条件可以嵌套，可以访问任意数目的关系。

下面是一些例子：

删除 Perryridge 分支机构的所有帐户

```
delete from account
```

```
where branch-name = "Perryridge"
```

删除位于 Needlham 的每一个分支机构的所有帐户

```
delete from
```

```
where branch-name in
```

```
(select branch-name
```

```
from branch
```

```
where branch-city = "Needlham")
```

◆ 删除余额低于银行平均余额的帐户记录

```
delete from account
```

```
where balance < (select avg(balance)
```

```
from account)
```

### 3.5.2 插入记录 (insert)

要在关系中插入数据，可以指定被插入的元组（单个或多个），或者是一条查询语句，该语句的查询结果是希望被插入的元组集合。显然，插入元组的属性值必须在属性域中。下面是几个例子：

假设要插入的信息是 Perryridge 分支机构中余额为 \$ 1200 的帐户 A-9732，可写成：

```
insert into account(branch-name, account-number, balance)
```

```
values("Perryridge", "A-9732", 1200)
```

这里，插入元组值的个数、排列顺序与关系的结构完全对应，所以 account 后面的属性序列可以省略。

插入元组中只有部分属性被赋值，所以关系 account 后的元组序列不可省略，



---

而且其余缺省的属性将被赋为 null.

如果想要插入多个指定的元组,可不必用多个 insert 语句,而是用“表”表达式来处理。例如,往关系 depositor 中连续插入三个元组,可表示为:

```
insert into depositor
(table (Hayes,A-102)
(Jones,A-217)
(Joy, A-715))
```

更多的情况是在查询结果的基础上执行插入。假设, Perryridge 分行想给每个在该行贷款的客户赠送一个 200 的新存款帐户,并以贷款号作为新存款帐户的帐户,可写作:

```
insert into account
select branch-name, loan-num, 200
from loan
where branch-name="Perryridge"
```

在这里,我们先执 select 子句,选出所有要插入的元组,再来执行 insert 子句。这个顺序是非常重要的。

### 3.5.3 更新记录 (update)

当需要修改关系中元组部分属性的值时,可用 update 语句实现。其结构如下:

```
update R
set column1=A1,column2=A2...
where F
```

其意义是:修改关系 R 中满足条件表达式 F 的元组中的指定属性值,修改的值在 set 子句中给出。含条件表达式 F 的 where 子句,可以是任何类型合法的 where 子句。

例如,想要执行“为所有存款大于平均数的帐户增加 5%的利息”可以写为:

```
update account
set balance=balance*1.05
where balance>select avg(balance)
from account
```

---

和 delete, insert 语句类似, 首先根据 update 语句中嵌套的 select—from—where 语句检查所有元组是否应该被更新, 然后才执行更新。

### 3.5.4 实验步骤

1. 恢复数据库, 将数据库备份 stud 恢复到 SQLSERVER 中
2. 执行 SQL 命令完成实验内容

1) 将院系中, 原院系名 “IS” 改为 “Information ”

```
Update dep set name=' Information' where name =' IS'
```

2) 在选课表中, 删除计算机科学与工程系学生选修 2 号课程的记录

```
Delete sc where cid=' 2' and sid in (select sid from stud join dep on  
stud.depid=dep.depid where depname = '计算机科学与工程' )
```

3) 在选课表中, 删除软件工程系学生选课 1 号课程的纪录记录

```
Delete sc where cid=' 1' and sid in (select sid from stud join dep on  
stud.depid=dep.depid where depname = '软件工程' )
```

4) 登记考试成绩:

```
Update sc set score=99 where sid=' 01' and cid=' 01'
```

.....

.....

5) 学号为 2406010103 的同学由原来的计算机科学与工程系转入信息安全系, 学号更改为 2406030102, 在数据库中做出相应修改。

```
Insert into stud(sid,sname...) select '2406030102',sname... from stud where  
sid=' 2406010103' ;
```

```
Update sc set sid=' 2406030102'' where sid=' 2406010103' ;
```

```
Delete stud where sid=' 2406010103' ;
```

### 3. 备份数据库

---

## 第四章 简单查询、多表查询

实验学时：2 学时

### 4.1 实验原理

结构化查询语言、表的连接、关系运算

### 4.2 实验目的

练习用 SELECT 查询语句，设置查询条件，实现单表查询。练习使用 SELECT 语句从多个表中查询数据，表的内连接、左外连接、右外连接的使用以及设置连接条件，理解连接条件和查询条件的在目的和功能上的区别。

### 4.3 实验内容与要求

- (1) 查询年龄在 20—22 之间的学生姓名（通过出生日期和当前日期计算年龄）
- (2) 查询年龄在 20—22 之间的学生姓名、院系和年龄
- (3) 查询所有教师的信息
- (4) 查询所有副教授的信息
- (5) 查询姓“张”的学生的学号、姓名、邮件地址
- (6) 查询所有有成绩（成绩不为空）的学生学号和课程号
- (7) 查询每个学生及其选修课程的情况
- (8) 查询选修了 2 号课程成绩在 60 分以下的所有学生 的学号、姓名、学生的邮件地址、课程名、教师姓名和教师的邮件地址及课程成绩
- (9) 查询选修了“数据库”的学生学号和姓名及教师姓名
- (10) 查询既选修了 1 号课程，又选修了 2 号课程的学生学号

### 4.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP

数据库：MS SQLSERVER

---

## 4.5 实验指导

### 4.5.1 表的连接

连接类型：

内连接, inner join, 或简写成 join

左(外)连接, left outer join, left join

右(外)连接, right outer join, right join

语法: table\_A join table\_B  
on table\_A.colName=table\_B.colName

例:

```
Select sid,sname,cid  
From stud join selectCourse on stud.sid=selectCourse.cid;
```

### 4.5.2 实验步骤

1. 恢复数据库, 将数据库备份 stud 恢复到 SQLSERVER 中
2. 执行 SQL 命令完成实验内容
  - 1) 查询年龄在 20—22 之间的学生姓名(通过出生日期和当前日期计算年龄, 方法见第八章)

```
Select sname from stud where year(getdate())-year(birthd) between 20 and  
22
```

- 2) 查询年龄在 20—22 之间的学生姓名、院系和年龄
- 3) 查询所有教师的信息

```
Select * from teacher
```

- 4) 查询所有副教授的信息

```
Select * from teacher where title =' 副教授'
```

- 5) 查询姓“张”的学生的学号、姓名、邮件地址

```
Select sid,sname,email from stud where sname like '张%'
```

- 6) 查询所有有成绩(成绩不为空)的学生学号和课程号

```
Select sid,cid from sc where score is not null
```

- 7) 查询每个学生及其选修课程的情况

---

```
Select stud.sid,sname,course.cname,score from stud join sc on  
stud.sid=sc.sid join course on sc.cid=course.cid
```

8) 查询选修了 2 号课程成绩在 60 分以下的所有学生 的学号、姓名、学生的邮件地址、课程名、教师姓名和教师的邮件地址及课程成绩

```
Select stud.sid,sname,stud.email,cname,tname,teacher.email,sc.score  
from stud join sc on stud.sid=sc.sid join course on sc.cid=course.cid join  
teacher on sc.tid=teacher.tid where sc.score <60
```

9) 查询选修了“数据库”的学生学号和姓名及教师姓名

```
Select stud.sid,stud.sname,teacher.tname from sc join stud on  
sc.sid=stud.sid join course on sc.cid=course.cid join teacher on  
sc.tid=teacher.tid where course.cname like ‘数据库’
```

10) 查询既选修了 1 号课程，又选修了 2 号课程的学生学号

```
Select sid from sc a join sc b on a.sid=b.sid where a.cid=' 1' and b.cid='  
2'
```

## 第五章 分组统计查询

实验学时：2 学时

### 5.1 实验原理

结构化查询语言、分组查询、集函数

### 5.2 实验目的

练习使用组函数 count(), max(), min(), avg() 等在 SQL 命令中实现统计功能。使用 GROUP BY 子句实现分组查询，以及组函数在分组查询中的应用。体会分组查询的功能特点。

### 5.3 实验内容与要求

- (1) 查询选修数据库课程的人数
- (2) 求每个学生的选课的门数，显示学号和选课门数

- 
- (3) 求每个学生选课的总学分数，显示学号和学分
  - (4) 求每个学生的总成绩，显示学号和总成绩
  - (5) 查询选修数据库并成绩在 60 分以上的人数
  - (6) 查询获得“数据库”课程最高分的学生姓名及成绩
  - (7) 求每门课程的平均成绩，并显示课程名及平均成绩
  - (8) 求每门课程的学生选修人数，并显示课程名及选修人数
  - (9) 求选修了 5 门以上课程的学生姓名及邮件地址

## 5.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP

数据库：MS SQLSERVER

## 5.5 实验指导

### 5.5.1 分组查询

有时候不仅希望将组函数作用在单个元组集上，而且也希望将其作用在一组元组集上。SQL 中可以用 group by 子句实现这个愿望。在 group by 子句中的一个或多个属性是用来构造分组的。group by 子句中所有属性有相同值的元组放在一个组中。

如，查询“找出每个分支机构的帐户结算平均额”，该查询书写如下：

```
select    branch-name, avg (balance)
from      account
group    by    branch-name
```

象上例这样，在计算平均值时保留重复元组是很重要的。而有些情况下在计算组函数前需先删掉重复元组，使用关键词 distinct。如“找出每个分支机构储户数”，在该例中不论一个客户有几个帐户，作为一名储户只计算一次，查询就该这样书写：

```
select branch-name, count (distinct    customer-name)
from    depositor,    account
where   depositor. account-number    =
account. account- number
```

### 5.5.2 分组查询条件

---

与 where 子句能在查询中选择和排除单个元组的功能一样, having 子句也可用于选择和排除一个元组集。而且有时, 对分组限定条件比对元组限定有用。例如, 我们也许只对帐户平均结算大于\$1200 的分支机构感兴趣。该条件并不是针对单个元组, 而是针对 group by 子句形成的分组。为了表达这样的查询, 使用 SQL 的 having 子句。 having 子句中的谓词在形成分组后才起作用, 因此可以使用组函数。用 SQL 表达该查询如下:

```
select  branch-name,  avg (balance)
from    account
group   by    branch-name
having  avg  (balance)>1200
```

如果在同一个查询中同时存在 where 子句和 having 子句, 那么首先应该用 where 子句中的条件表达式。满足 where 条件表达式的元组通过 group by 子句形成分组。 having 子句若存在, 就将作用于每一分组。

### 5.5.3 组函数

组函数是以一个值集合为输入, 返回单个值的函数。SQL 提供了六个预定义组函数:

- 求某一列值的平均值: avg ( )
- 求某一列值的最小值: min ( )
- 求某一列值的最大值: max ( )
- 求某一列值的总和: sum ( )
- 计算元组个数: count ( )
- 计算某一列值的个数: count ( \* )

组函数的自变量可以是简单的字段名, 也可以是 SQL 表达式。

例如, 求出销售人员的销售量占销售目标的百分比。

```
select      avg(100*(sales/quota))
from        salesreps
```

sum 和 avg 的输入必须是数字, 而其他函数还可作用在非数字数据类型如字符串、日期 / 时间上。

我们经常使用组函数 count 计算一个关系中数据值或元组的个数。

如, 求资产超过 \$ 2100000 的银行分行个数。

```
select  count(assets)
from    branch
where   assets>$ 2100000
```

注意 count ( ) 函数忽略了数据项的值, 只是简单计算有多少数据项。所以, 它实际上不关心你指定那个字段作为 count ( ) 函数的自变量。

因此, SQL 支持一个特殊的组函数: count ( \* ). 该函数计算行 (元组)

---

而不是数据值。所以，上面的查询可重写为

```
select    count (*)
from      branch
where     assets>$2100000
```

SQL 不允许在用 count(\*)时使用 distinct。在用 max 和 min 时使用 distinct 是合法的，尽管结果是一样的。

SQL 中是不允许对组函数进行复合的。因而，如 max(avg(...)) 等是不允许的。

#### 5.5.4 实验步骤

1. 恢复数据库，将数据库备份 stud 恢复到 SQLSERVER 中

2. 执行 SQL 命令完成实验内容

1) 查询选修数据库课程的人数

```
Select count(*) from sc join course on sc.cid=course.cid group by cid where
cname like '数据库'
```

2) 求每个学生的选课的门数，显示学号和选课门数

```
Select sid,count(cid) from sc group by sid
```

3) 求每个学生选课的总学分数，显示学号和学分

```
Select sid,sum(cridits) from sc group by sid
```

4) 求每个学生的总成绩，显示学号和总成绩

```
Select sid,sum(score) from sc group by sid
```

5) 查询选修数据库并成绩在 60 分以上的人数

```
Select count(*) from sc join course on sc.cid=course.cid group by cid where
cname like '数据库' and score >60
```

6) 查询获得“数据库”课程最高分的学生姓名及成绩

```
select mc.cname,mc.ms,stud.sname,teacher.tname from
(select course.cid cid,cname,max(score) ms from sc join course on
course.cid=sc.cid
group by course.cid,cname having cname like '数据库%') as mc
join sc on mc.cid=sc.cid and mc.ms=sc.score
join stud on sc.sid=stud.sid
join teacher on sc.tid=teacher.tid
```

7) 求每门课程的平均成绩，并显示课程名及平均成绩

```
Select cname,avg(score) from course join sc on sc.cid=course.cid group by
cid
```

8) 求每门课程的学生选修人数，并显示课程名及选修人数

```
Select cname,count(*) from sc join course on sc.cid=course.cid group by
cname
```



---

9) 求选修了 5 门以上课程的学生姓名及邮件地址

```
Select sname,email from stud join sc on stud.sid=sc.sid group by sid having  
count (*) >=5
```

---

## 第六章 集合操作、子查询

实验学时：2 学时

### 6.1 实验原理

结构化查询语言、集合运算、子查询

### 6.2 实验目的

IN、EXISTS、NOT EXISTS 运算在 WHERE 子句中的应用；静态集合和由 SELECT 命令产生的动态结果集运算。

### 6.3 实验内容与要求

- (1) 查询其他系中比信息系(depId=' IS' )某一学生年龄小的学生姓名和年龄
- (2) 查询没有选修任何课程的学生姓名、所在院系及邮件地址
- (3) 查询选修了全部课程的学生姓名
- (4) 查询既选修了 1 号课程，又选修了 2 号课程的学生姓名

### 6.4 实验器材（设备、元器件）

操作系统：Windows 2000/XP

数据库：MS SQLSERVER

### 6.5 实验指导

#### 6.5.1 实验步骤

子查询是嵌套在一个 select 语句中的另一个 select 语句。当需要从一个表中检索信息，

---

检索条件值又是来自该表本身的内部数据时，子查询非常有用。

子查询可以嵌入以下 SQL 子句中：where 子句、having 子句和 from 子句。

例：查询工资比编号为 7566 雇员工资高的雇员姓名。

```
select ename
from emp
where sal >
      (select sal
       from emp
       where empno=7566)
order by ename;
```

说明：

- (1) 子查询要用括号括起来；
- (2) 将子查询放在比较运算符的右边；
- (3) 不要在子查询中使用 order by 子句，select 语句中只能有一个 order by 子句，并且它只能是主 select 语句的最后一个子句。

## 1、单行子查询

内部 select 语句只返回一行结果的查询（单列）。主查询的 where 子句使用单行子查询返回结果要采用单行比较运算符（=、>、>=、<、<=、<>）。

### ● Where 子句中使用单行子查询

例：显示和雇员 scott 同部门的雇员姓名、工资和部门编号。

```
select ename, sal, deptno from emp
where deptno= (select deptno from emp where ename='SCOTT');
```

### ● 单行子查询中使用组函数

---

例：显示工资最低的雇员姓名、工作和工资。

```
select ename, job, sal
from emp where sal=(select min(sal) from emp);
```

- **having 子句中使用单行子查询**

例：显示部门内最低工资比 20 部门最低工资要高的部门的编号及部门内最低工资。

```
select deptno as 部门编号, min(sal) as 最低工资
from emp
group by deptno
having min(sal)>(select min(sal)
                  from emp where deptno=20);
```

## 2、多行子查询

内部 select 语句返回多行结果,主查询的 where 子句使用多行子查询返回的结果要采用多行比较运算符,多行比较运算符可以和一个或多个值进行比较。

多行运算比较符: in、any、all

- **使用 in 运算符的多行子查询**

In 运算符将等于列表中的任意一项。

例 1: 查询有下属的雇员姓名、工作、工资和部门号。

```
select ename, job, sal, deptno
from emp
where empno in (select mgr from emp);
```

思考?

如果要查询没有下属的雇员姓名、工作、工资和部门号,如下的 SQL 语句是否可以获得预期的结果?如果不能,应如何修改?

---

```
select ename, job, sal, deptno
from emp
where empno not in (select mgr from emp);
```

子查询返回的结果中有一个 mgr 是空值，not in 运算符将会用主查询条件（empno）与子查询中的每个结果（mgr）进行逻辑非的比较。因为子查询返回结果中有条空值，任何条件和空值比较都是空值。因此只要空值成为子查询的一部分，就不能用 not in 运算符。

SQL 语句更正如下：

```
select ename, job, sal, deptno
from emp
where empno not in (select mgr from emp where mgr is not null);
```

#### ● 使用 any 运算符的多行子查询

Any 运算符将和内部查询返回的结果逐个比较，与单行操作符配合使用。

<any: 表示比子查询返回结果中的最大值小；

=any: 表示可以是子查询返回结果中的任意一个值；

>any: 表示比子查询返回结果中的最小值大。

例: 查询工资低于某个文员（CLERK）雇员工资，但不从事文员工作的雇员编号、姓名、工种和工资。

```
select empno, ename, job, sal
from emp
where sal < any (select sal from emp where job='CLERK')
and job <> 'CLERK';
```

#### ● 使用 all 运算符的多行子查询

All 运算符将和内部查询返回的每个结果比较。

>all: 比最大的大；

<all: 比最小的小。

---

例：查询高于所有部门平均工资的雇员姓名、工作、工资和部门编号。

```
select ename, job, sal, deptno
from emp
where sal > all (select avg(sal) from emp group by deptno);
```

### 3、多列子查询

多列子查询返回多列结果的内部 select 语句，多列子查询中的列比较有成对比较与不成对比较两种方法。

多列子查询分为成对比较多列子查询和非成对比较多列子查询。

#### ● 成对比较多列子查询

例：查询与部门编号为 30 的部门中任意一个雇员的工资和奖金完全相同的雇员姓名、工资、奖金、部门编号，满足该雇员不是来自 30 号部门。

(1) 查询 30 部门内雇员工资和奖金

```
select sal, comm from emp where deptno=30;
```

(2) 查询非 30 部门内雇员姓名、工资、奖金和部门编号

```
select ename, sal, comm, deptno from emp where deptno <> 30;
```

(3) 把 (1) 作为 (2) 的子查询

查询 (2) 中与查询 (1) 中工资和奖金完全匹配的只有 SMITH 一个雇员，下面找出该员工。

```
select ename, sal, comm, deptno
from emp
where deptno <> 30
      and (sal, comm) in (select sal, comm
                           from emp
                           where deptno=30);
```

#### 3.2 非成对比较多列子查询

例：查询工资与 30 部门中任意一个雇员的工资相等，同时奖金也与 30 部门中任意一个雇员奖金相等的雇员姓名、工资、奖金、部门编号，但该雇员不是来自 30 号部门。

---

```
select ename, sal, comm, deptno
from emp
where deptno<>30
      and sal in (select sal
                  from emp
                  where deptno=30)
      and comm in (select comm
                  from emp
                  where deptno=30);
```

### 3、相关子查询 (exists)

相关子查询指需要引用主查询列表的子查询语句，通过 exists 谓词实现。对主查询的每条记录都需执行一次子查询来测试是否匹配。若子查询返回结果非空，则主查询的 where 子句返回值为 true，否则返回值为 false。

例：查询在纽约（NEW YORK）工作的雇员姓名、工种、工资和奖金。

```
select ename, job, sal, comm
from emp
where exists (select *
             from dept
             where emp.deptno=deptno and loc='NEW YORK');
```

### 4、from 子句中使用子查询

在 from 子句中使用子查询时，必须给子查询指定别名。

例：显示工资高于部门平均工资的雇员姓名、工作、工资和部门号。

```
select ename, job, sal, emp.deptno
```

---

```
from emp, (select deptno, avg(sal) avgsal
           from emp
           group by deptno) s
where emp.deptno=s.deptno and sal>s.avgsal;
```

练习：查询各部门中工资等级最高的雇员姓名、工作、工资、工资等级和部门号。

方法一、from 子句中使用一个子查询

```
select e.ename, e.job, e.sal, s.grade, e.deptno
from emp e,
     salgrade s,
     (select max(s.grade) grade, e.deptno
      from emp e, salgrade s
      where e.sal between s.losal and s.hisal
      group by e.deptno) q
where e.sal between s.losal and s.hisal
     and e.deptno=q.deptno and s.grade=q.grade
order by e.deptno;
```

方法二、from 子句中使用两个子查询

```
select p.ename, p.job, p.sal, p.grade, p.deptno
from (
     select e.ename, e.job, e.sal, s.grade, e.deptno
     from emp e, salgrade s
     where e.sal between s.losal and s.hisal) p,
     (
     select max(s.grade) grade, e.deptno
     from emp e, salgrade s
     where e.sal between s.losal and s.hisal
     group by e.deptno) q
where p.deptno=q.deptno and p.grade=q.grade
```



---

order by p.deptno;

### 6.5.2 实验步骤

1. 查询其他系中比信息系 (depid='IS') 某一学生年龄小的学生姓名和年龄
  - a) `Select sname,age from stud where age < any (select age from stud where depid='IS')`
  - b) `Select sname,age from stud where age < (select min(age) from stud where depid='IS')`
2. 查询没有选修任何课程的学生姓名、所在院系及邮件地址
  - a) `Select sname,depname,email from (stud inner join dep on stud.depid=dep.depid) where not exist (select cid from sc where sc.sid=stud.sid)`
  - b) `Select sname,depname,email from (stud inner join dep on stud.depid=dep.depid) where sid not in (select distinct sid from sc)`
3. 查询选修了全部课程的学生姓名
  - a) `Select sname from stud where not exeists (select cid from course where cid not in (select cid from sc where sc.sid=stud.sid))`
4. 查询既选修了 1 号课程, 又选修了 2 号课程的学生学号
  - a) `select sid from sc where cid='1') and sid in (select sid from sc where cid='2')`
5. 查询既选修了 1 号课程, 又选修了 2 号课程的学生姓名
  - a) 使用集合运算 in 和自连接方法实现  
`Select sname from stud where sid in (select a.sid from (sc a join sc b on a.cid=1 and b.cid=2 and a.cid=b.cid) )`
  - b) 采用集合运算 in :  
`Select sname from stud where sid in (select sid from sc where cid='1') and sid in (select sid from sc where cid='2')`
  - c) 采用集合运算 intersect  
`...(select sid from sc where cid='1') intersect (select sid from sc where cid='2')`

---

---

## 第七章 数据库建模

实验学时：4 学时

### 7.1 实验原理

使用图形化 CASE 工具设计数据库，ER 图

### 7.2 实验目的

本实验要求学生学习数据库建模工具 PowerDesigner 的使用方法，掌握最基本的使用方法。本实验将重点练习：

使用 PDM 生成数据库

导出数据库脚本

利用 PD 的逆向工程生成 PDM 并进行修改

### 7.3 实验内容与要求

使用 PDM，以图形化界面方式创建表及确定各表之间的关系；

由生成的数据库导出数据库的脚本；

根据生成的脚本，再利用 PD 的逆向工程生成 PDM 并进行修改。

### 7.4 实验设备与环境

操作系统：Windows

数据库：SQLSERVER UDB

应用软件：Power Designer 9.5

### 7.5 实验指导

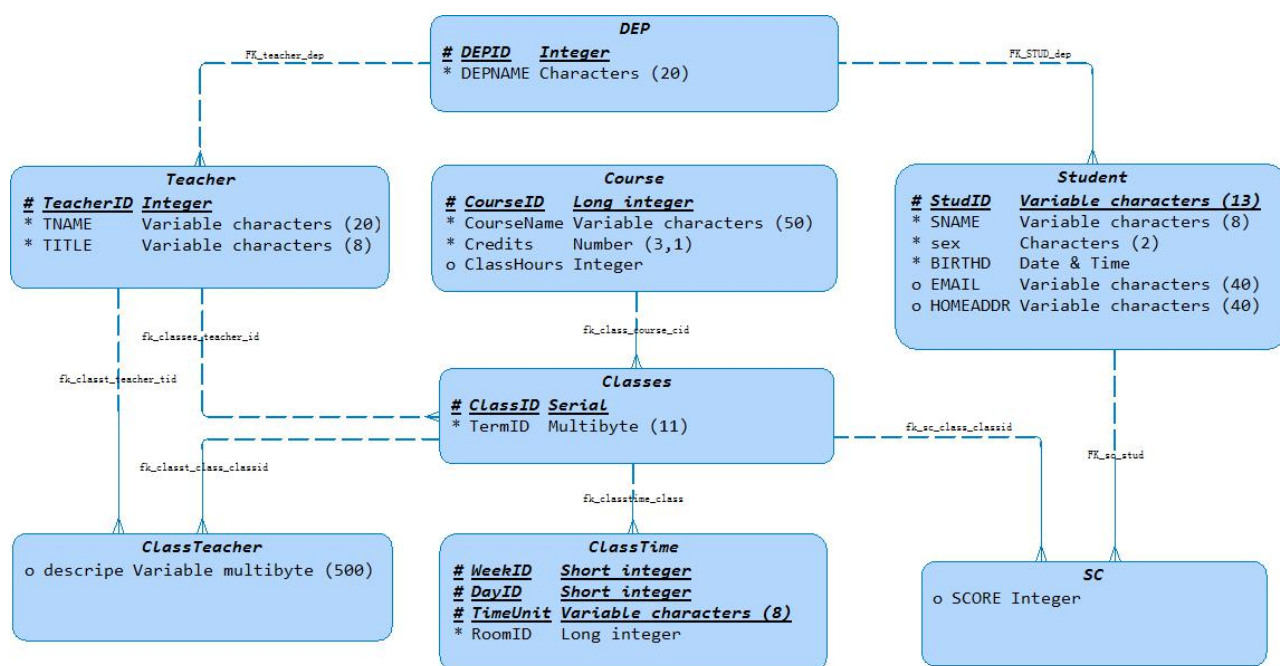
#### 7.5.1 Power Designer

详见下一章.....

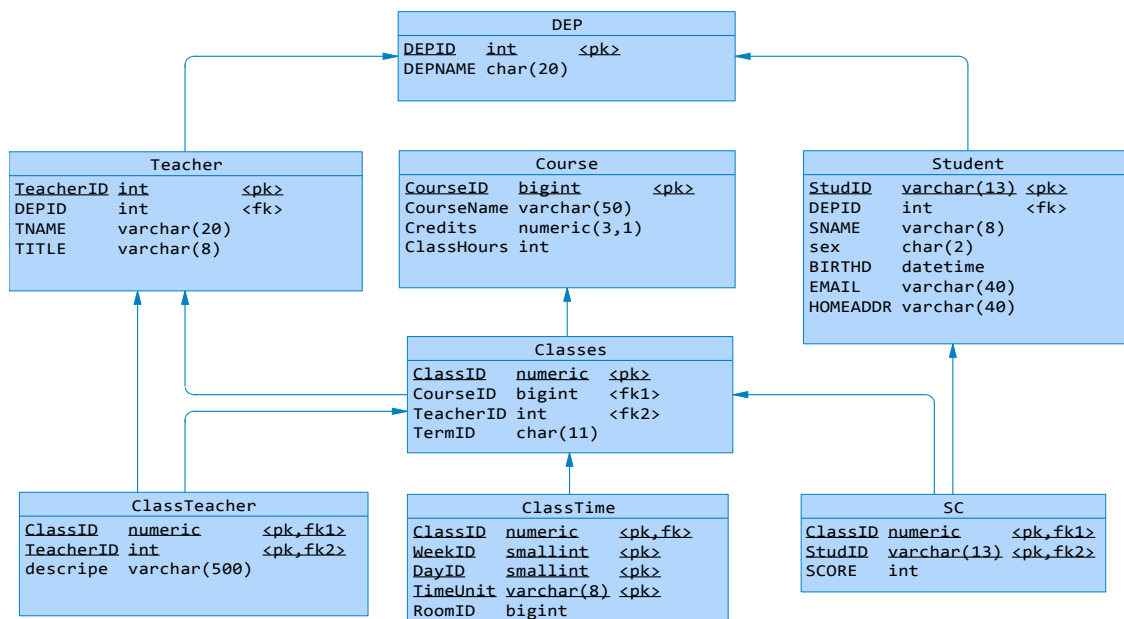
#### 7.5.2 实验步骤

1. 通过生成 Physical Data Model (PDM) 以图形化界面创建表及确定各表之间的关系。

使用相关软件自主设计，创建概念模型：



转换成物理模型：



## 2. 根据第一步生成的表导出数据库的脚本

生成数据库代码:

```

/*=====*/
/* Table: course */
/*=====*/
create table course
(
    cidchar(8)    not null,
    cname        varchar(20),
    pcid         char(8),
    sscore       decimal(2,1),
    primary key (cid)
);

/*=====*/
/* Table: depart */
/*=====*/
create table depart
(
    depart       varchar(20) not null,
    primary key (depart)
);
  
```

---

```

/*=====*/
/* Table: jianli */
/*=====*/
create table jianli
(
    sidchar(10)    not null,
    date1    date        not null,
    date2    date        not null,
    addr     varchar(20),
    "work"   varchar(100),
    primary key (sid, date1, date2)
);

/*=====*/
/* Index: Relationship_7_FK */
/*=====*/
create index Relationship_7_FK on jianli (
sid ASC
);

/*=====*/
/* Table: sc */
/*=====*/
create table sc
(
    tidchar(10)    not null,
    cidchar(8)     not null,
    sidchar(10)    not null,
    score    decimal(4,1),
    primary key (tid, cid, sid)
);

/*=====*/
/* Index: Relationship_4_FK */
/*=====*/
create index Relationship_4_FK on sc (
sid ASC
);

```

---

```

/*=====*/
/* Index: Relationship_5_FK */
/*=====*/
create index Relationship_5_FK on sc (
tid ASC
);

/*=====*/
/* Index: Relationship_6_FK */
/*=====*/
create index Relationship_6_FK on sc (
cid ASC
);

/*=====*/
/* Table: student*/
/*=====*/
create table student
(
    sidchar(10)    not null,
    dep_departvarchar(20),
    sname        char(10),
    depart       varchar(20),
    birthd       date,
    email        varchar(20),
    addr         varchar(20),
    primary key (sid)
);

/*=====*/
/* Index: Relationship_2_FK */
/*=====*/
create index Relationship_2_FK on student (
dep_depart ASC
);

/*=====*/
/* Table: teacher*/

```

---

```

/*=====*/
create table teacher
(
    tidchar(10)    not null,
    dep_departvarchar(20),
    tname    char(10),
    title    char(10),
    depart    varchar(20),
    email    varchar(20),
    phone    char(20),
    birthd    date,
    basesalarydecimal(8,2),
    salary2    decimal(8,2),
    primary key (tid)
);

/*=====*/
/* Index: Relationship_3_FK */
/*=====*/
create index Relationship_3_FK on teacher (
    dep_depart ASC
);

alter table jianli
    add foreign key FK_JIANLI_RELATIONS_STUDENT (sid)
        references student (sid)
        on update restrict
    ;

alter table sc
    add foreign key FK_SC_RELATIONS_STUDENT (sid)
        references student (sid)
        on update restrict
    ;

alter table sc
    add foreign key FK_SC_RELATIONS_TEACHER (tid)
        references teacher (tid)
        on update restrict

```



---

```

;

alter table sc
  add foreign key FK_SC_RELATIONS_COURSE (cid)
    references course (cid)
    on update restrict
;

alter table student
  add foreign key FK_STUDENT_RELATIONS_DEPART (dep_depart)
    references depart (depart)
    on update restrict
;

alter table teacher
  add foreign key FK_TEACHER_RELATIONS_DEPART (dep_depart)
    references depart (depart)
    on update restrict
;
```

### 3. 利用 PD 的逆向工程生成 PDM 并进行修改

---

## 第八章 Powerdesinger 操作指南

### 8.1 PowerDesigner 简介

PowerDesigner 是 Sybase 公司的 CASE 工具集，使用它可以方便地对管理信息系统进行分析设计，它几乎包括了数据库模型设计的全过程。利用 PowerDesigner 可以制作数据流程图、概念数据模型、物理数据模型，可以生成多种客户端开发工具的应用程序，还可为数据仓库制作结构模型，也能对团队设计模型进行控制。它可与许多流行的数据库设计软件，例如：PowerBuilder, Delphi, VB 等相配合使用来缩短开发时间和使系统设计更优化。

#### 8.1.1 PowerDesigner 安装

PowerDesigner 在 windows 系统下的安装比较简单，跟普通应用程序安装没有什么区别，在这里就不再进行介绍了，大家可以自己安装一下就可以体会到了。

PowerDesigner 主要包括以下几个功能部分：

##### 8.1.1.1 DataArchitect

这是一个强大的数据库设计工具，使用 DataArchitect 可利用实体-关系图为一个信息系统创建“概念数据模型”-CDM (Conceptual Data Model)。并且可根据 CDM 产生基于某一特定数据库管理系统（例如：Sybase System 11）的“物理数据模型”-PDM (Physical Data Model)。还可优化 PDM，产生为特定 DBMS 创建数据库的 SQL 语句并可以文件形式存储以便在其他时刻运行这些 SQL 语句创建数据库。另外，DataArchitect 还可根据已存在的数据库反向生成 PDM, CDM 及创建数据库的 SQL 脚本。

##### 8.1.1.2 ProcessAnalyst

这部分用于创建功能模型和数据流图，创建“处理层次关系”。

- AppModeler：为客户/服务器应用程序创建应用模型。
- ODBC Administrator：此部分用来管理系统的各种数据源。

#### 8.1.2 PowerDesigner 的 4 种模型文件：

---

#### **8.1.2.1 概念数据模型 (CDM)**

CDM 表现数据库的全部逻辑的结构, 与任何的软件或数据储藏结构无关。 一个概念模型经常包括在物理数据库中仍然不实现的数据对象。 它给运行计划或业务活动的数据库一个正式表现方式。

#### **8.1.2.2 物理数据模型 (PDM)**

PDM 叙述数据库的物理实现。 借由 PDM , 你考虑真实的物理实现的细节。 它进入账户两个软件或数据储藏结构之内拿。 你能修正 PDM 适合你的表现或物理约束。

#### **8.1.2.3 面向对象模型 (OOM)**

一个 OOM 包含一系列包, 类, 接口 , 和他们的关系。 这些对象一起形成所有的 ( 或部份) 一个软件系统的逻辑的设计视图的类结构。 一个 OOM 本质上是软件系统的一个静态的概念模型。

你使用 PowerDesigner 面向对象模型建立面向对象模型. (OOM) 你能为纯粹地对象- 导向的靠模切目的建立一个 OOM, 产生 Java 文件或者 PowerBuilder 文件, 或你能使用一个来自 OOM 的物理数据模型 (PDM) 对象 , 来表示关系数据库设计分析。

#### **8.1.2.4 业务程序模型 (BPM)**

BPM 描述业务的各种不同内在任务和内在流程, 而且客户如何以这些任务和流程互相影响。 BPM 是从业务合伙人的观点来看业务逻辑和规则的概念模型, 使用一个图表描述程序, 流程, 信息和合作协议之间的交互作用。

以下是其中的 CDM、PDM 和 OOM 三者的转换关系

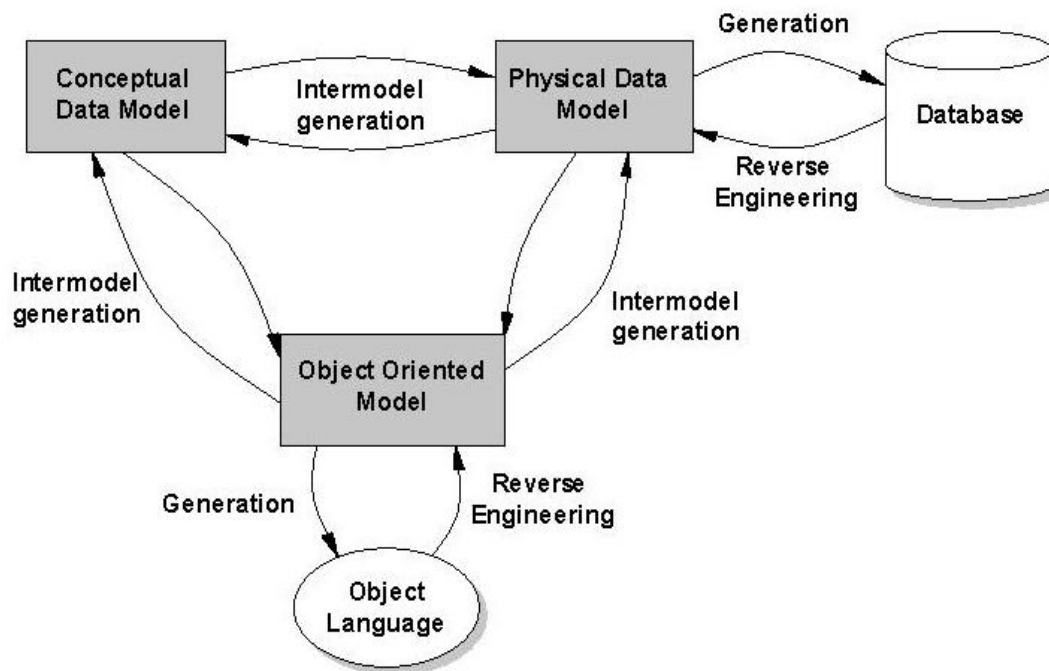


图 1

## 8.2 使用 PowerDesigner 环境

PowerDesigner 的主窗口主要由四部分组成：对象浏览器（对象浏览器可以用分层结构显示你的工作空间）；输出窗口（显示操作的结果）；结果列表（用于显示生成、覆盖和模型检查结果，以及设计环境的总体信息。）；图表窗口（用于组织模型中的图表，以图形方式显示模型中各对象之间的关系）。

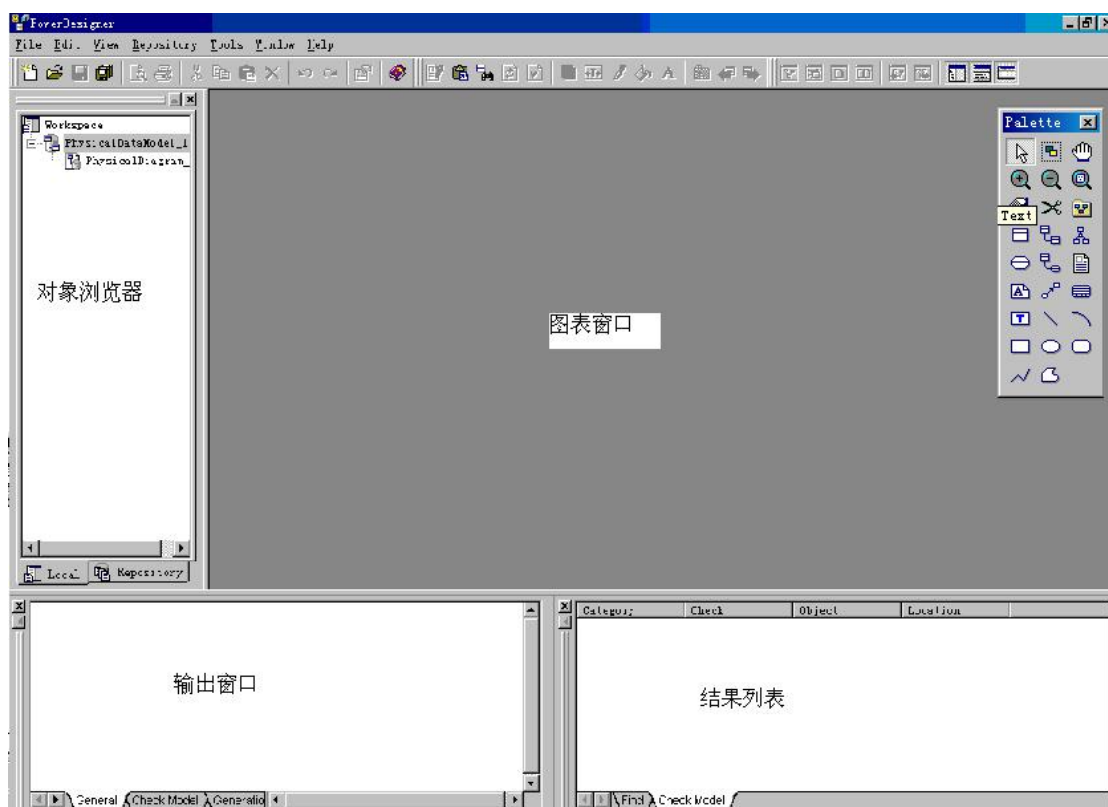


图 2

### 8.2.1 PD 界面和基本的操作方式

PD 主要的界面主要是由下面几部分组成的：

- 树形模型浏览器，对象浏览器可以用分层结构显示你的工作空间；
- 输出窗口，显示操作的结果；
- 结果列表，用于显示生成、覆盖和模型检查结果，以及设计环境的总体信息；
- 图表窗口，用于组织模型中的图表，以图形方式显示模型中各对象之间的关系；
- 其他的窗口与其他的软件差不多，在这里就不介绍了。具体界面如图 3 所示：

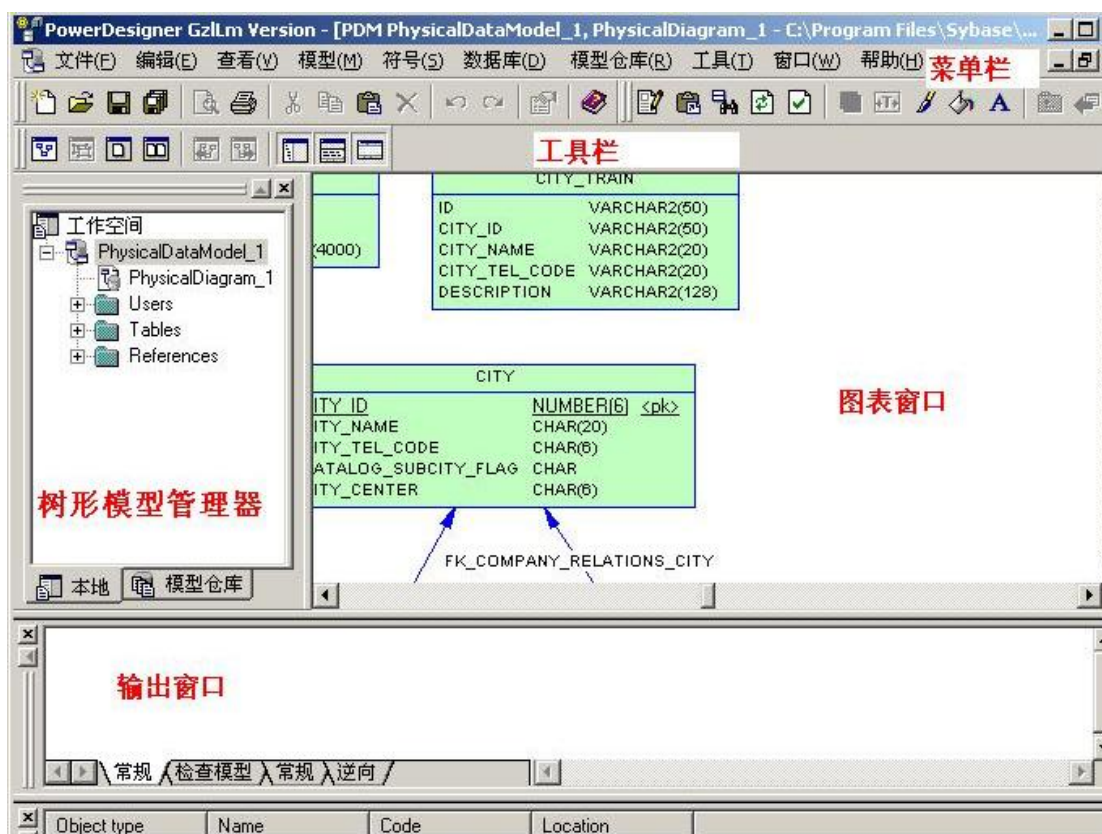


图 3

下面举例创建一个概念模型来看看 PD 的操作方式

1. 按工具面板（就是 Palette 工具集）的实体工具。
2. 当光标移动进图表的时候，变成实体的形状。

3. 在 CDM 图表中点击任何一处。一个实体符号在点击位置出现。实体名字为 Entity\_n, n 是一个创建对象的次序编号。
4. 实体工具仍然是可使用的, 重复第三步在 CDM 图表中产生另外的一个实体。现在有 CDM 图表的二个实体。
5. 点击工具面板的关系工具。这时, 实体工具被现在释放, 而且关系工具是可使用的。
6. 点击在第一个实体之内而且当继续按着鼠标按钮的时候, 拖拉光标到第二个实体。在第二个实体之内放开鼠标按钮。这样可以产生关系。通过右击关系图标, 可以打开该关系的属性并进行编辑, 如图 4 所示:

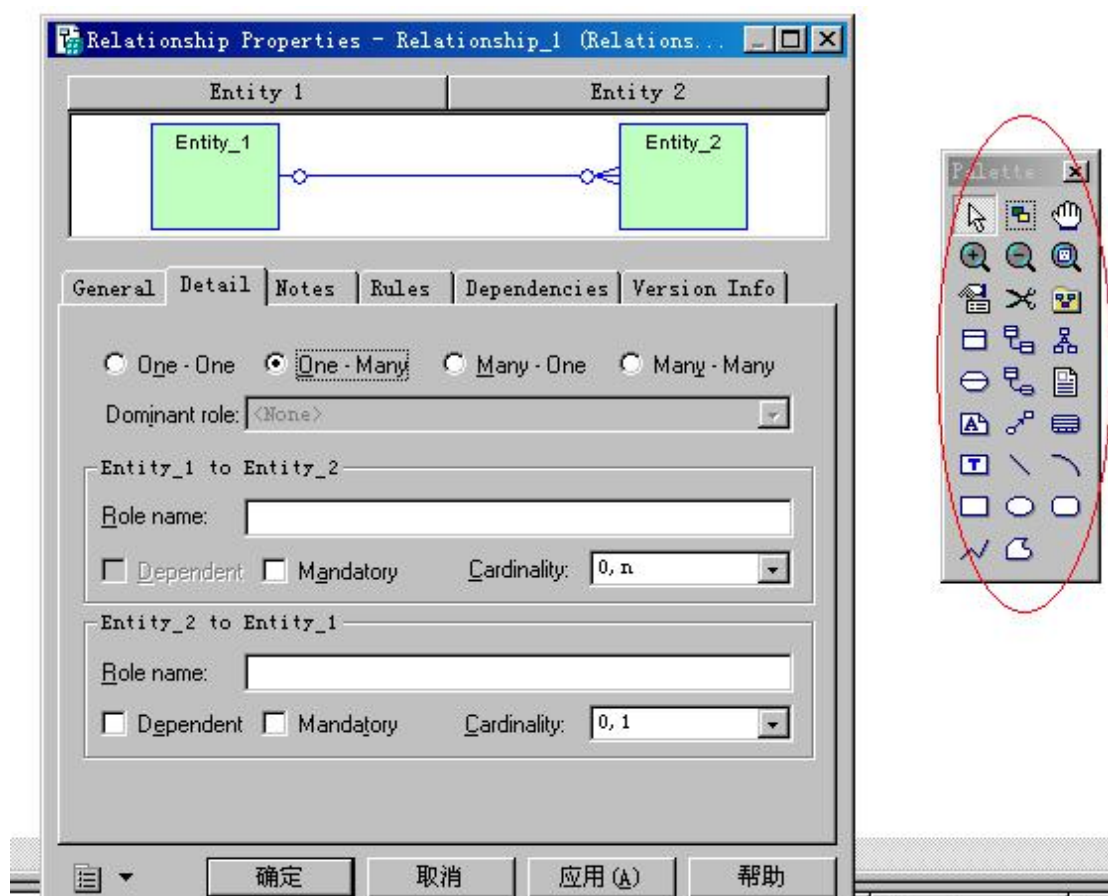


图 4

7. 点击鼠标右键, 释放关系工具。一个工具保持可使用直到释放它。释放一个工具, 可以选择另外的一个工具或按鼠标右键。默认的, 当按鼠标右键, 指针工具被激活。

- 
8. 点击面板的套索工具，套索工具是现在可使用。
  9. 在第一个实体的上面角落点击光标，按着鼠标按钮，拖拉光标拉一个包括两个实体的长方形，放开鼠标按钮，实体和关系被选择。
  10. 拖拉实体到一个新位置，关系跟随实体一起移动。
  11. 点击面板的文本工具。文本工具是现在可使用。
  12. 在关系下面点击光标，一些文本在被长方形指出的区域中出现。
  13. 点击鼠标右键，你释放文本工具。
  14. 双击文本，一个文本框出现。
  15. 在文本框中输入短文本。
  16. 点击 OK，文本在图表中出现。
  17. 点击文本框的一个柄，按着鼠标左键，拖拉光标到右边直到所有的文本出现，放开鼠标按钮，在图表背景上点击，文本框柄消失。
  18. 点击面板的指针工具。你将使用这个工具选择并且删除符号。
  19. 在实体符号上点击，选择你想删除的对象
  20. 按键盘上的 DEL 键，确认信息对话框出现，问你如何删除选择。如图 5 所示。如果你选择删除对象，你将删除图标符号并且删除模型中的对象。如果你只选择删除符号，你将删除图标符号，但是保存模型的对象。

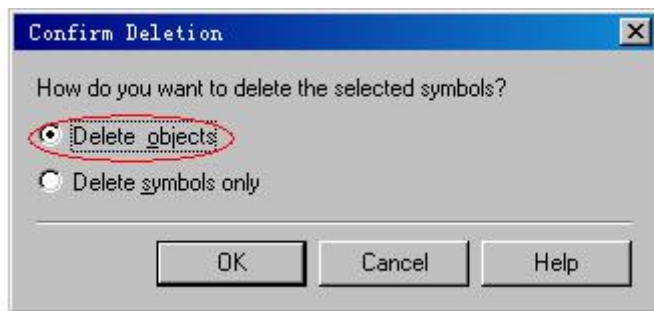


图 5

21. 点击 OK，图表中的实体和联合的关系被移动。对象也从模型删除。
22. 点击剩余的实体，当你点击文本的时候，按着键盘的 SHIFT 键，二个对象将被选择。
23. 按 DEL 键，并且在删除信息出现的时候点击 OK，剩余的实体和文本被删除。



## 8.2.2 PD 的概念模型工具和以及业务规则

### 8.2.2.1 工具图标

下面图 6 给出了在 PD 中一些常用的工具图标：






图形	名称	操作	图形	名称	操作
	指针	选择符号		联合连接	插入联合连接符号
	套索	一个区域的选择符号		文件	插入一个文件符号
	整体选择	选择全部符号，一起设置大小		注释	插入注释符号
	放大	放大视野范围		连接 / 扩展依赖	在图表中的符号之间画一个图形连接，在注释和一个对象之间画一个注释连接，在两个支持扩展依赖的对象间画一个扩展依赖
	缩小	缩小视野范围		主题	插入主题符号
	打开包图表	显示选择包的图表		文本	插入文本
	属性	显示选择的符号属性		线条	插入一条线
	删除	删除符号		圆弧	插入一个圆弧
	包	插入包符号		长方形	插入一个长方形
	实体	插入实体符号		椭圆	插入一个椭圆
	关系	插入关系符号		圆角矩形	插入一个圆角矩形
	继承	插入继承符号		折线	插入一条折线
	联合	插入联合符号		多边形	插入一个多边形

图 6

### 8.2.2.2 业务规则

业务规则是业务活动中必须遵循的规则,是业务信息之间约束的表达式,它反

映了业务信息数据之间的彝族完整性约束. 每当信息实体中包含的信息发生变化的时候, 系统都会检查这些信息是否违反特定的业务规则.

业务规则有的六种类型: 事实, 定义, 公式, 确认, 需求和约束, 具体概念如图 7 所示:

业务规则类型	业务规则说明	业务规则举例
事实	信息系统中存在的事实	一个出版者可能出版一或多个的主题的图书
定义	信息系统中对象的特性	一位作家被一个名字和一个住址识别
公式	信息系统中的计算公式	总金额为所有订单金额的总和
确认	信息系统中需要的确认	支付所有作家一本书的版税百分比必须为版税的 100%
需求	信息系统中功能的详细说明	模型被设计以致版税的总数量不超过总售卖的 10%
约束	信息系统中数据之间的约束	销售开始日期必须迟于出版日期

图 7

### 8.2.2.3 产生一条新的业务规则

你将会产生一条业务规则标明该如何将版税归还于作家。

1) 选择模型的业务规则 (菜单: Model→Business Rules)

业务规则的列表对话框显示已存在的业务规则。

2) 点击增加一行工具

一支箭头在新空行的开始处出现, 并且一个默认的名字和代码被输入。默认为定义类型的业务规则, 图 8 给出了添加业务规则的例子:



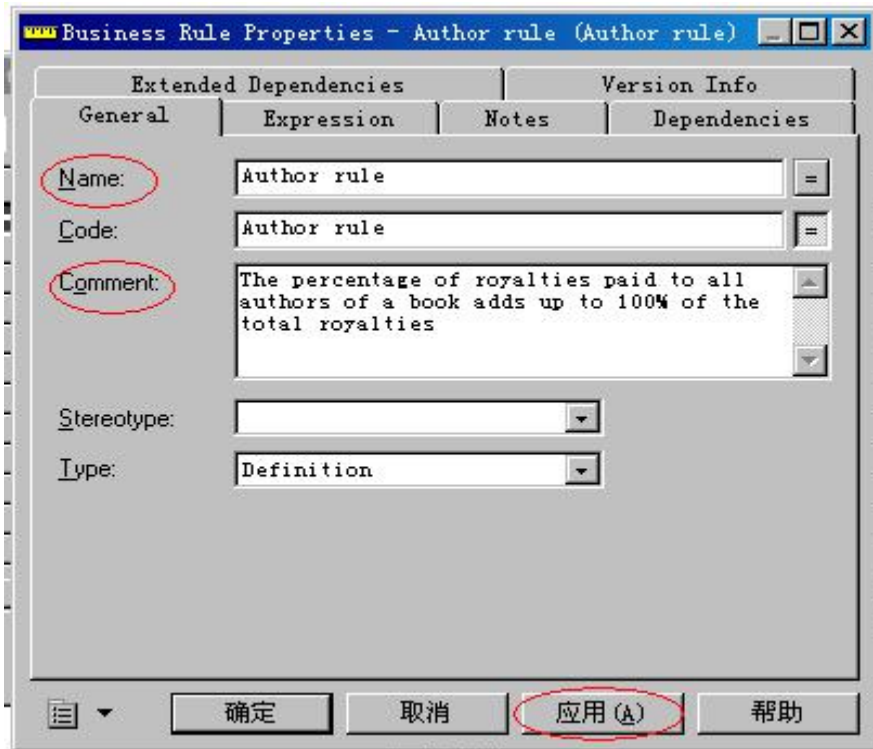


图 9

### 8.2.2.4 数据项目

数据项目的概念：一个数据项目是一个基本的信息。

### 8.2.2.5 创建一个新的数据项目

要管理多个作家的多个书，你将会对每位作家版税的百分比创建数据项目并且为作家列表中的的作家名称排序。

- 1) 选择菜单栏的模型数据项目。数据项目的列表显示已存在的数据项目（当然也有可能为空，即原来没有数据项目）。
- 2) 点击增加一行工具。 一个箭头在第一个空白行的开始和一个默认的名称出现，而且代码被进入。
- 3) 键入名称列的 TitleAuthor Percent 。 这是数据项目的名称。 相同的代码自动地在代码列中被输入。如下面图 10 所示：

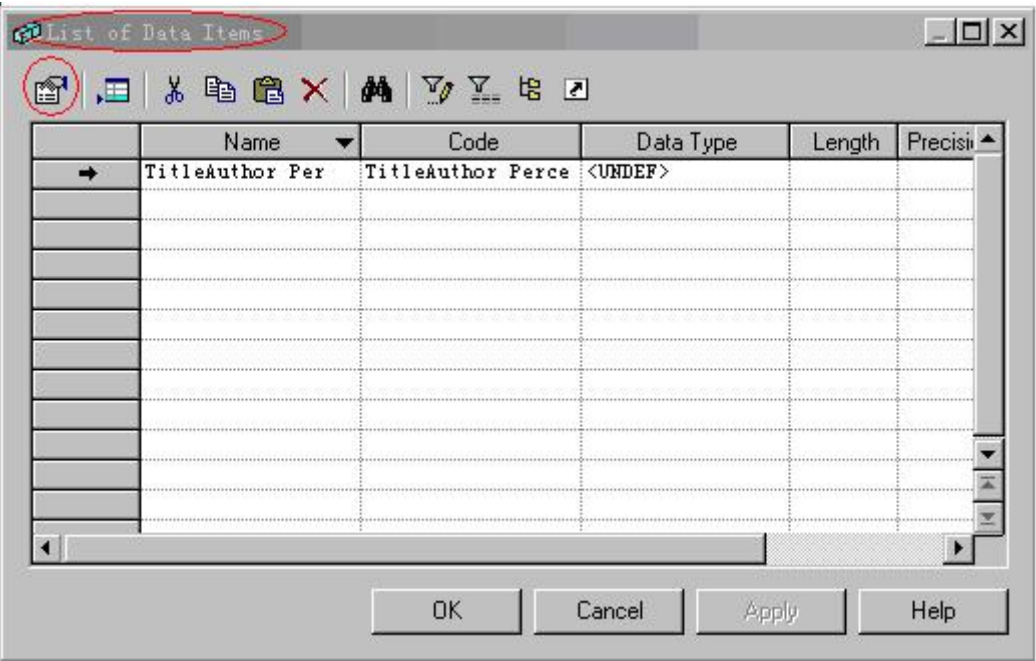


图 10

- 4) 点击应用。

新创建的数据项目的被提交。名称按字母顺序地分类, 当按应用或列表的确定的时候，所有的名称按字母顺序地被分类。列表的名称次序将会以其中任何一个操作而改变。

- 5) 点击新的数据项目行。一个箭头在行开始处出现。
- 6) 点击属性工具（图 10 的红圈所示），或在行开始处双击箭头。

属性页为新创建的数据项目。在数据类型列中，SI 指出短整数型。如图 11 所示：

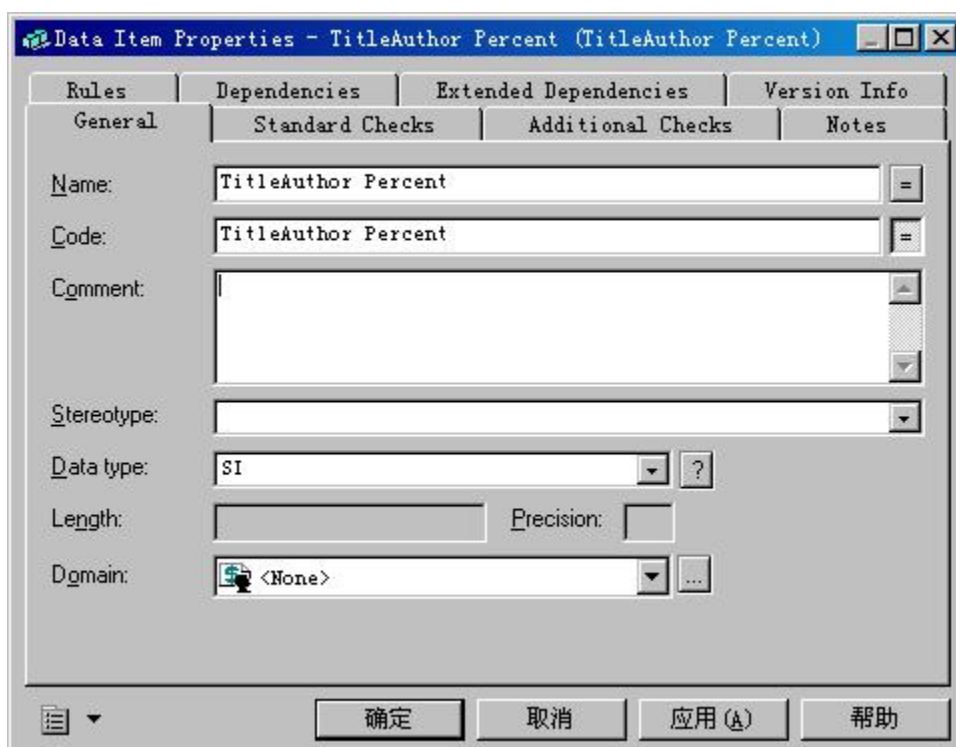


图 11

- 7) 点击确定便完成了一个数据项目的建立。

#### 8.2.2.6 删除一个数据项目

删除一个数据项目比较简单，就是在图 10 中选定要删除的数据项，然后点击图上方工具栏中的删除图标即可完成。

---

## 8.2.3 定义和使用继承

### 8.2.3.1 创建一个继承连接

你从子实体到父实体创建一个继承连接。你将会从 PERIODICAL 和 NONPERIODICAL 实体到 TITLE 实体定义一个继承。

- 1) 选择工具面板的继承工具。
- 2) 在 NONPERIODICAL 实体之内点击，按住鼠标左键不放, 拖拉光标到 TITLE 实体。在 TITLE 实体之内放开鼠标按钮。如图 12 所示：

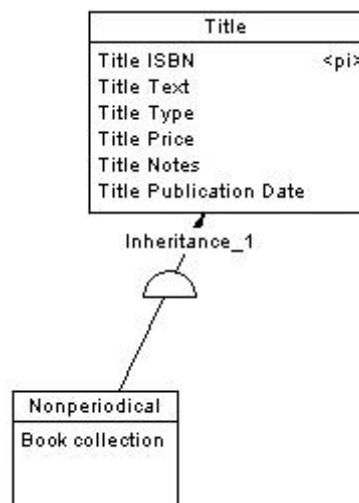


图 12

一个继承连接出现在这些实体之间。连接中间有一个半圆形，和一个指向 TITLE 父实体的箭头。NONPERIODICAL 是子实体。子实体继承它的父实体。

- 3) 选择工具板上的继承工具，点击半圆形，按住鼠标左键不放, 拖拉光标到 PERIODICAL 实体。在 PERIODICAL 实体内放开鼠标按钮，继承符号因此会改变，如图 13 所示：

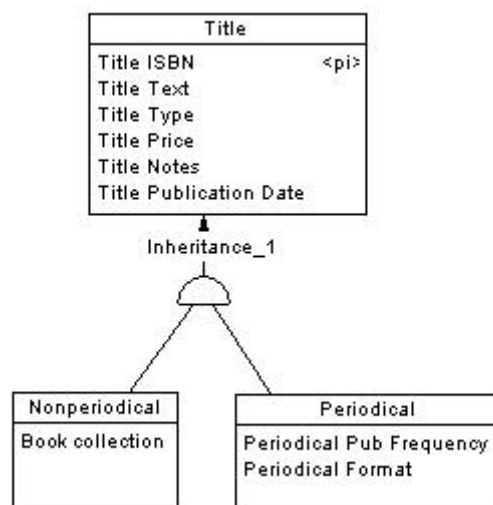


图 13

### 8.2.3.2 定义继承属性

我们可以分配一个主题到继承，使它互斥，而且定义它的生成模态。

- 1) 点击工具面板的指针工具。
- 2) 双击继承连接的中央半圆形。继承特性页出现。
- 3) 类型名称在名称框中继承。这是继承的名称。
- 4) 选择子对象互斥的复选框。因为主题是一份期刊或一份非期刊, 不能两个兼有，所以是互斥的。如图 14 所示：



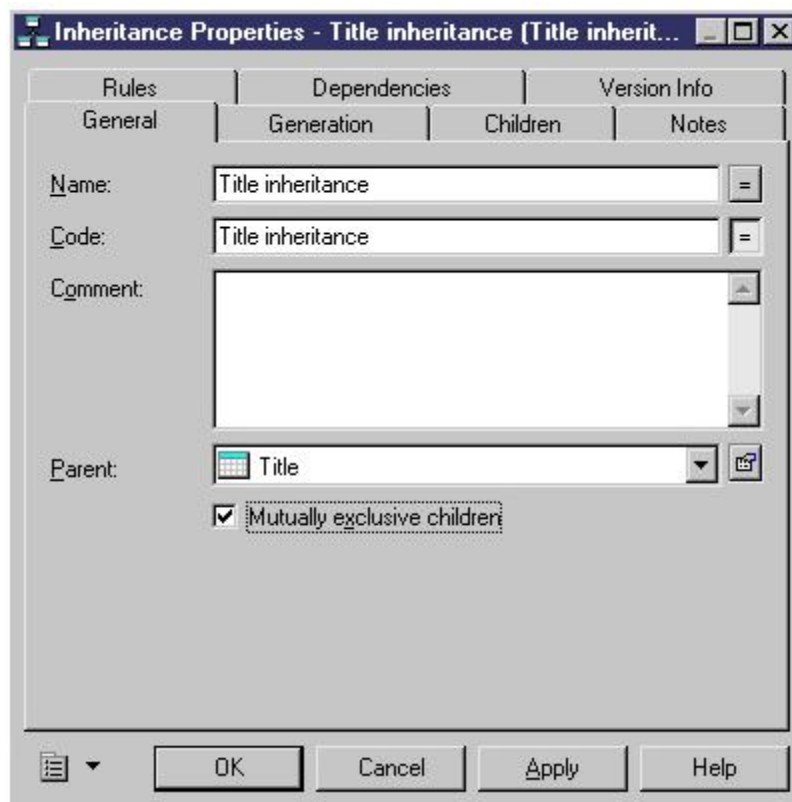


图 14

5) 点击生成定位键。生成页出现。

6) 在生成模式分组框中确定产生子对象复选框没有被选择。

这里的选择将影响继承如何产生物理数据模型 PDM。仅仅生成父对象意味着只有一个表将被生成，在这里例子中，你只需要知道在每个子对象中的属性是不同的。

7) 输入 Periodical 到名称列中，如图 15 所示：

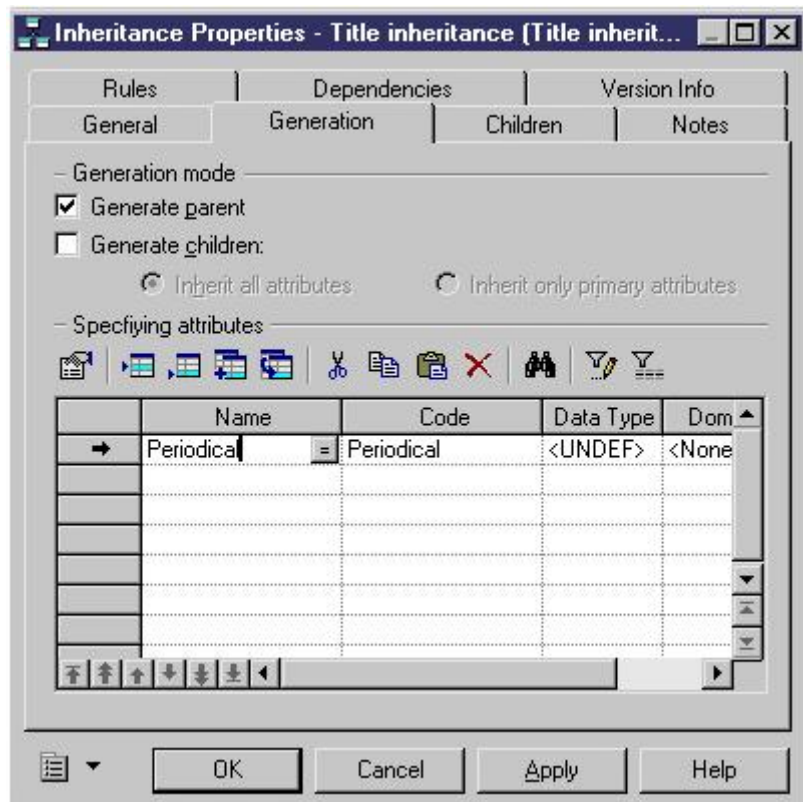


图 15

- 8) 在数据类型列中点击。
- 9) 点击下拉列表箭头。一个下拉列表框出现。
- 10) 选择来自下拉列表框的 BL，这时已为那个指定属性定义 Boolean(BL) 数据类型。如图 16 所示：

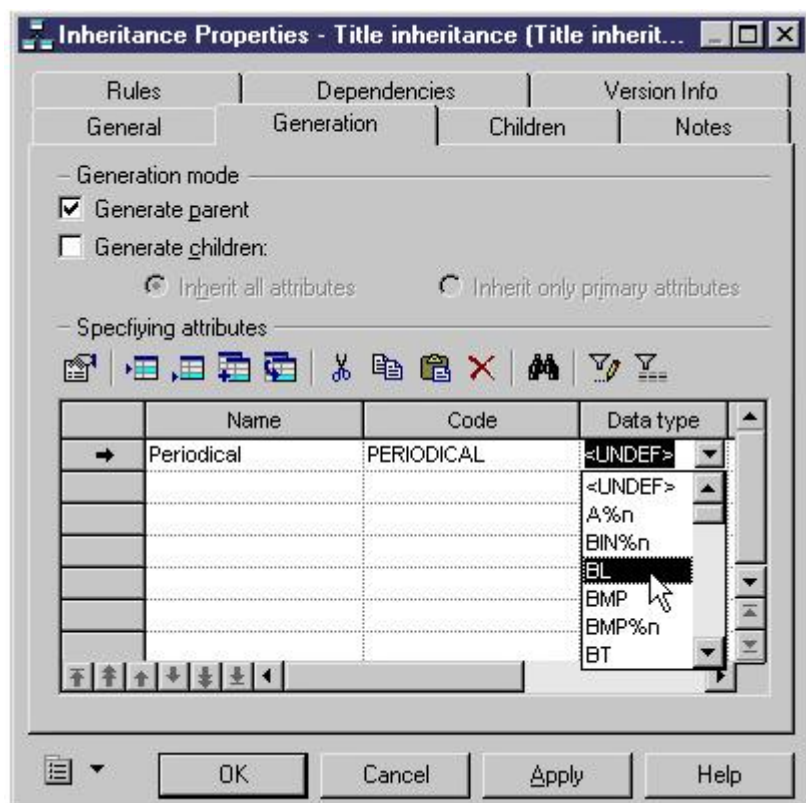


图 16

11) 移动到右边，知道出现 M 复选框。

这是强制性的复选框。当选择的时候，它指出当 Periodical 列被产生的时候不能包含 NULL 价值。

12) 选择 M 复选框。

13) 点击确定。这时两者之间的关系发生了变化，如图 17 所示：

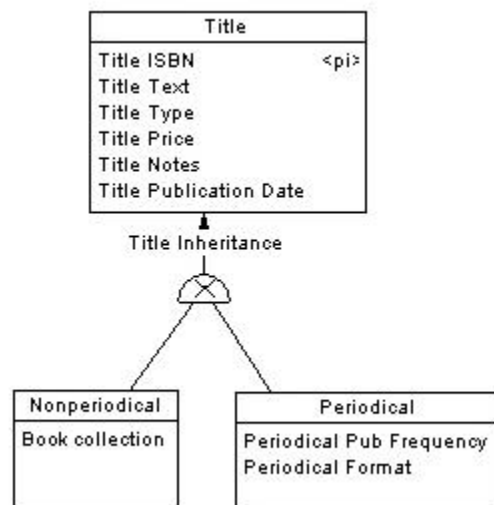


图 17

在图 17 中，十字架在半圆形中出现，指出继承是互斥的。继承不只是出现在编程中，在数据库设计中同样用到。

### 8.3 将 CDM 对象转换成 PDM 对象

下面给出了 CDM 对象转成 PDM 对象的步骤：

- 1) 选择 Tools—>Generate Physical Data Model。PDM 生成选项对话框出现。
- 2) 从数据库管理系统下拉列表框选择 Sybase AS Anywhere 8 。  
这个对话框处理前面生成的教程文件名称作为 PDM 文件的名称，但是扩展名为 PDM。
- 3) 选择 Share 单选按钮。

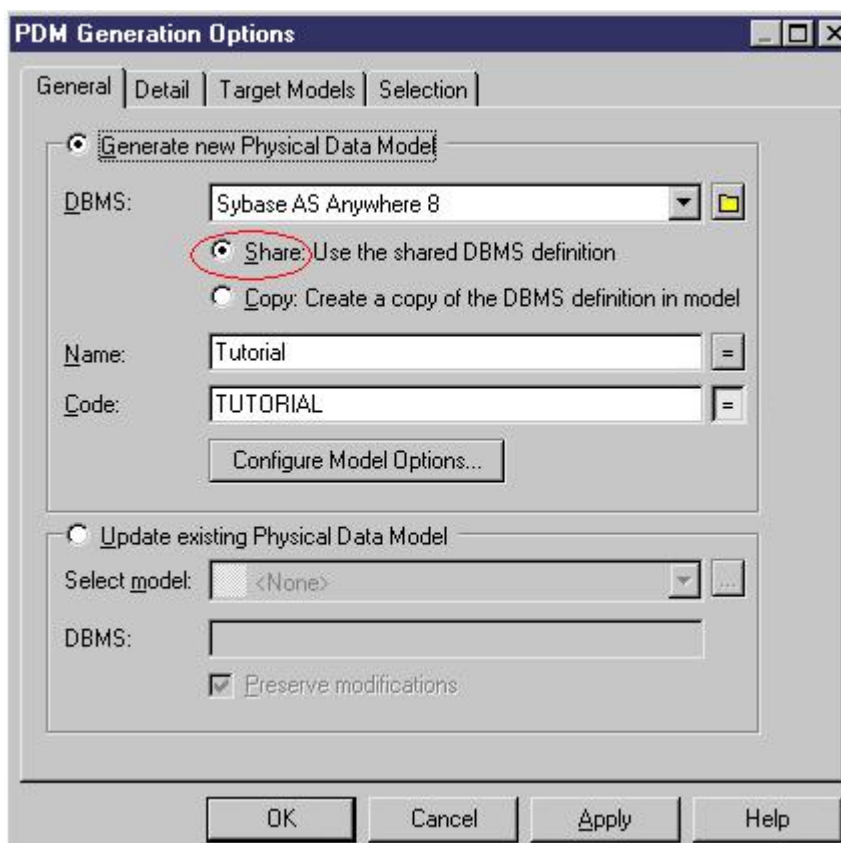


图 18

如图 18 所示，指出使用数据库管理系统库中储存的数据库管理系统定义文件。

- 4) 点击细节定位键。细节页出现。
- 5) 挑选出来的或清除下列各项选项，如图 19 所示：

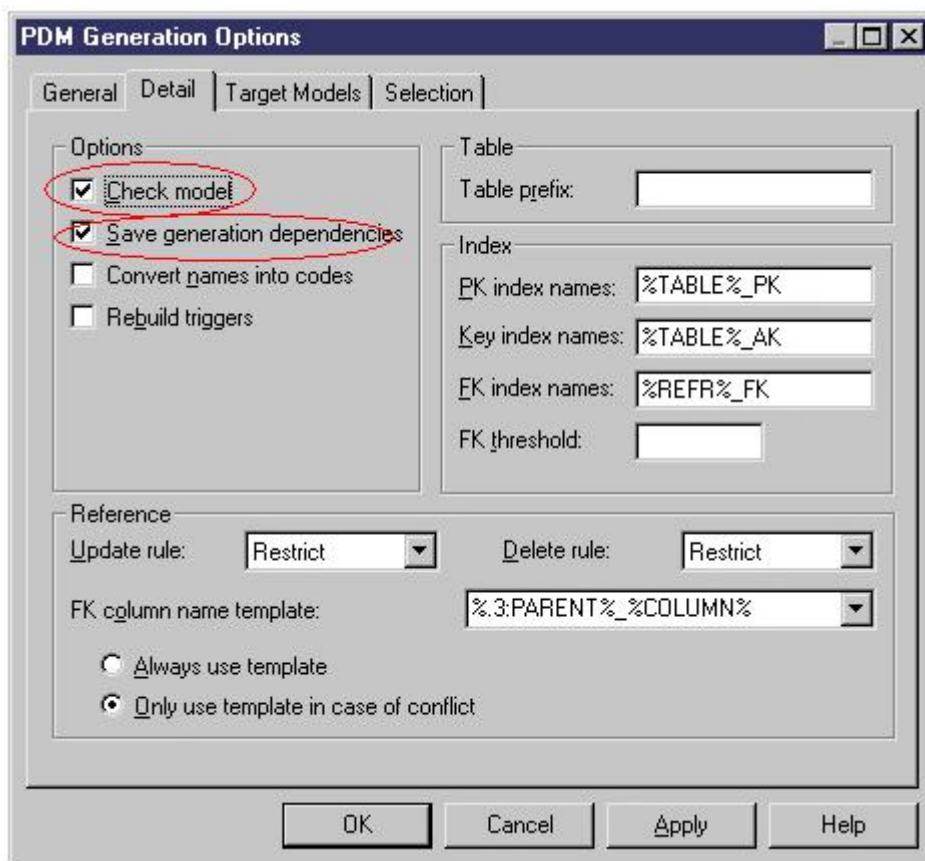
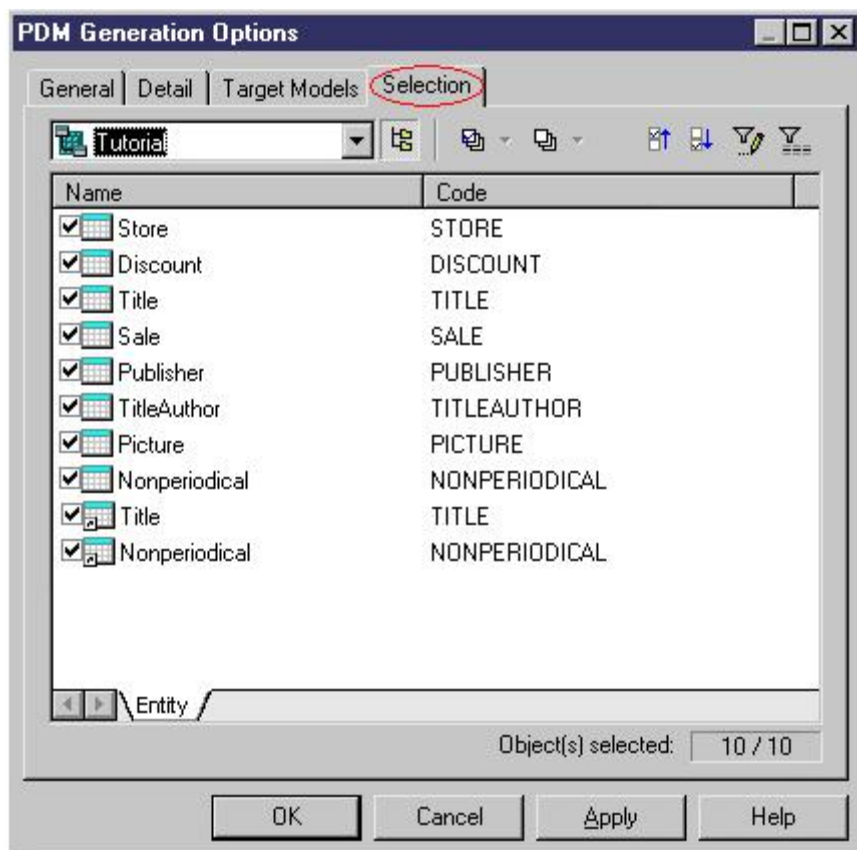


图 19

在图 19 中，如果我们选择了 Check Model，模型将会在生成之前被检查。Save Generation Dependencies 选项决定 PowerDesigner 是否为每个模型的对象保存对象识别标签，这个选项主要用于合并由相同 CDM 生成的两个 PDM。

- 6) 选择 Selection 定位键，它列出 CDM 的所有对象。默认地，所有的对象被选择，如图 20 所示：



- 7) 点击确定。
  - 8) 当程序已经完成生成时，关闭结果窗口。
- PDM 在模型的窗口中出现，如图 21 所示：

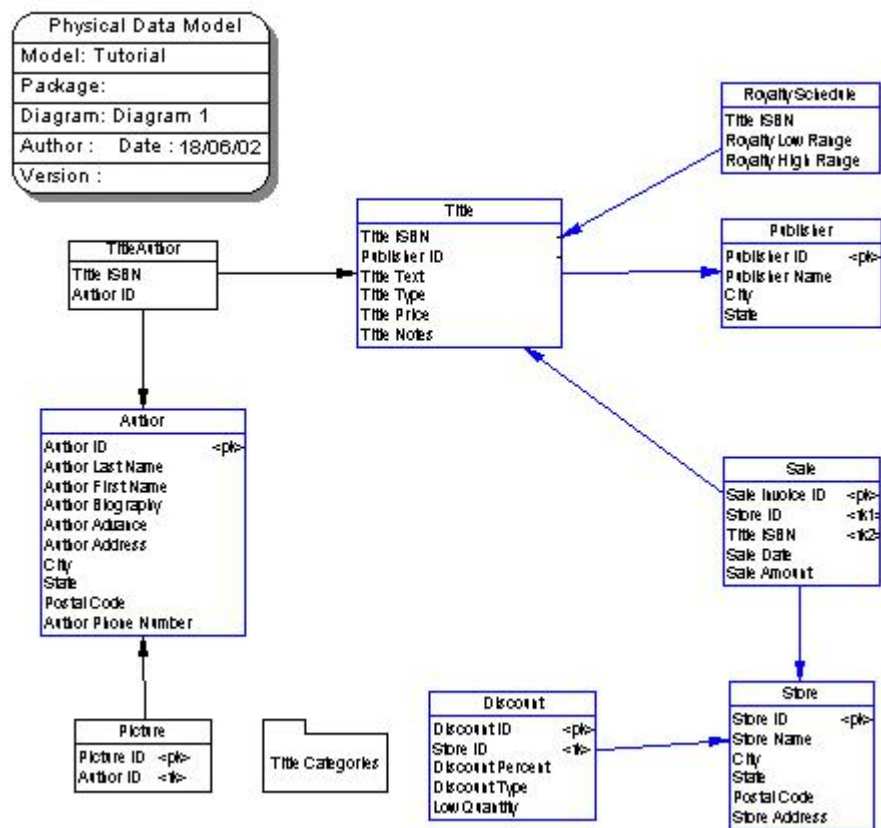


图 21



---

## 8.4 正向工程和逆向工程

### 8.4.1 正向工程

我们可以直接地从 PDM 产生一个数据库，或产生一个能在你的数据库管理系统环境中运行的数据库脚本，这是正向工程。

默认是生成与 PDM 相同数据库的脚本，但是也可以产生其他数据库的脚本  
产生一个数据库创建脚本

- (1) 选择 Database→Generate Database。数据库生成对话框出现。它显示生成参数。默认参数已经被选择。
- (2) SQL 的文件名称框中键入 PDM\_TUTORIAL。
- (3) 在目录框中, 输入一条路径。
- (4) 选择生成脚本的按钮。
- (5) 选择仅仅生成一个文件。
- (6) 点击 Selection 定位键。
- (7) 点击底部表定位键，表页列出模型中选择可用的所有数据库表。
- (8) 全部点击选择工具。这选择所有的表复选框。

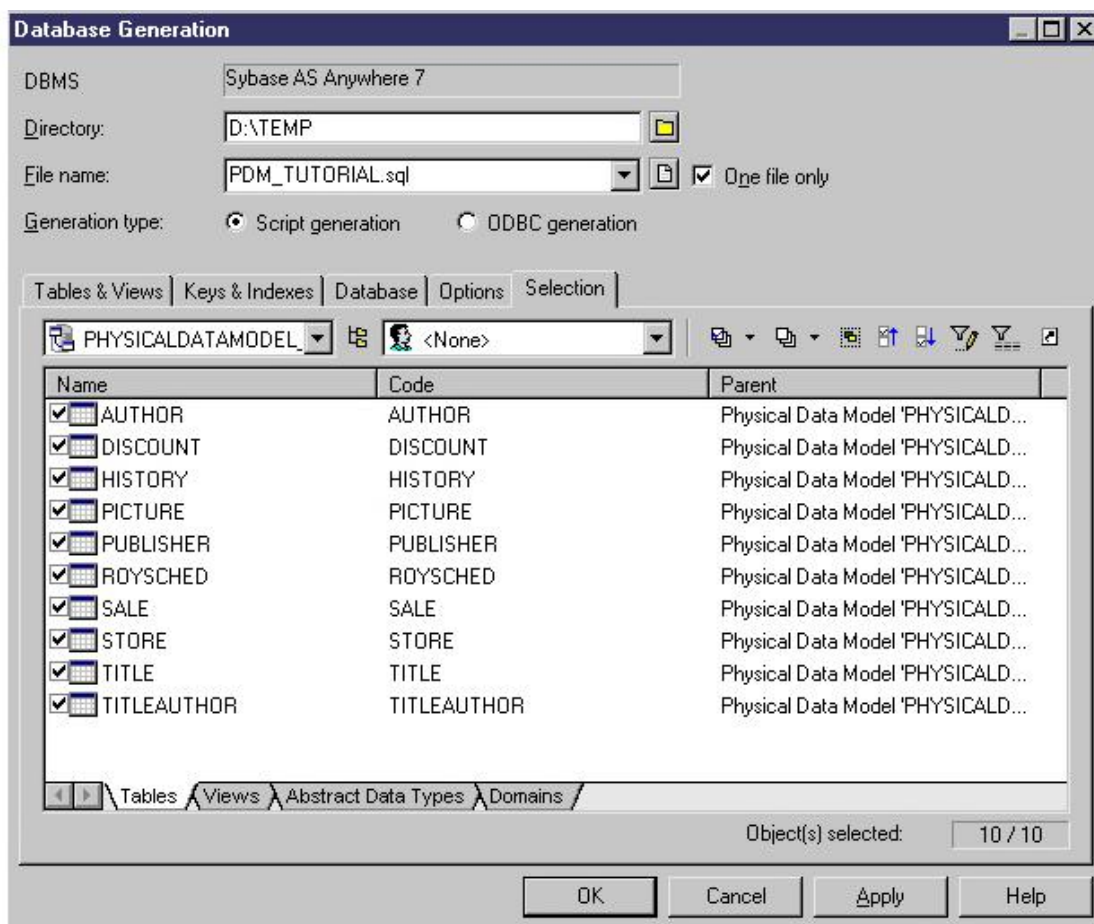


图 22

(9) 切换 Views 和 Domains 选择需要的视图和域。

(10) 点击确定。

可以生成数据库脚本, 如果选择 ODBC 方式, 则可以直接连接到数据库, 从而直接产生数据库表以及其他数据库对象。

## 8.4.2 逆向工程

逆向工程主要用于下列情况:

- 设计和实现的、已经存在的数据库没有建立数据模型。
- 数据模型不能反映现存数据库设计的实际情况。
- 现存数据库的数据模型丢失。
- 现存的数据库需要转换到不同的 RDBMS 上, 这时需要有不同的物理数据模型。

逆向工程数据库对象从一个脚本文件到新的 PDM，逆向工程来自一个脚本文件的数据库对象，下面是一个例子：

- (1) 选择 File->Reverse Engineer->Database 显示新的物理数据模型的对话框。
- (2) 点击部份 Radio 按钮。
- (3) 选择下拉列表框的一个数据库管理系统。

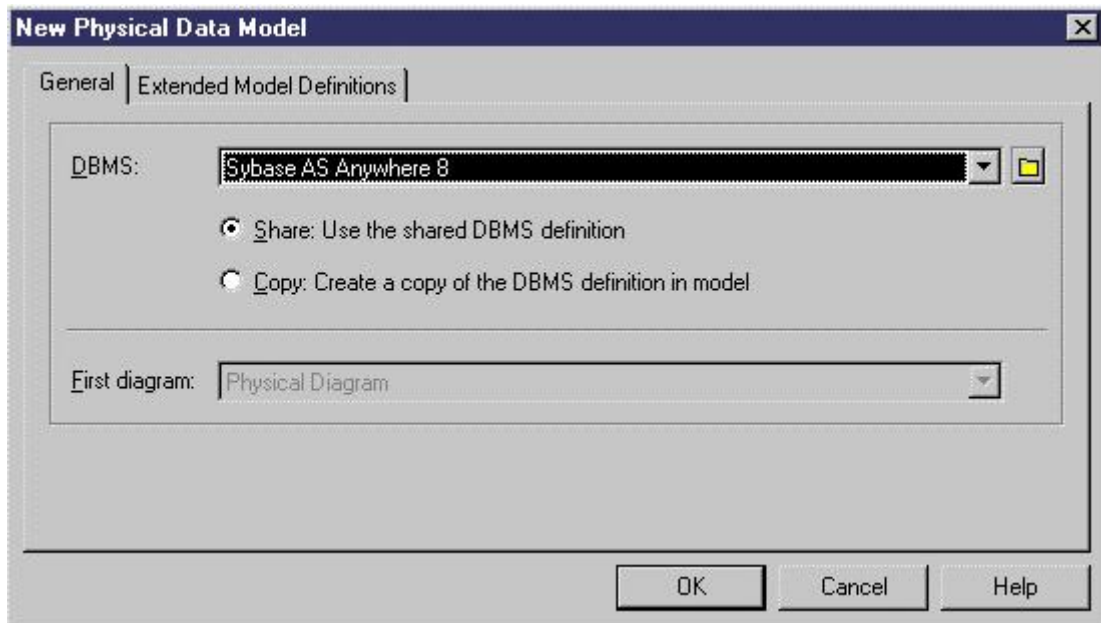


图 23

- (4) 点击确定。数据库逆向工程对话框出现。
- (5) 点击使用脚本文件的 Radio 按钮。
- (6) 浏览适当的目录选择脚本文件。

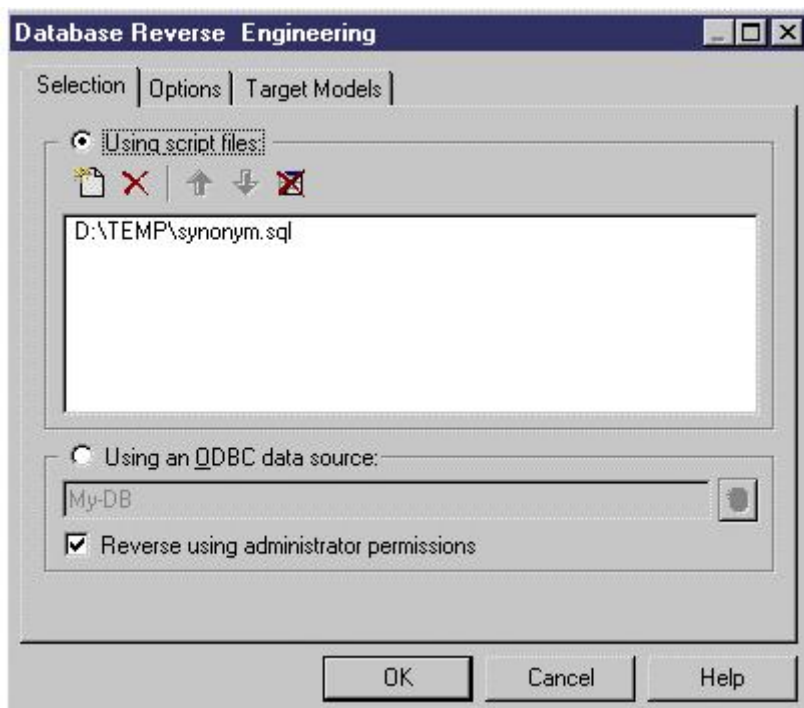


图 24

- (7) 点击选项定位键显示选项页。
- (8) 选择逆向工程选项。

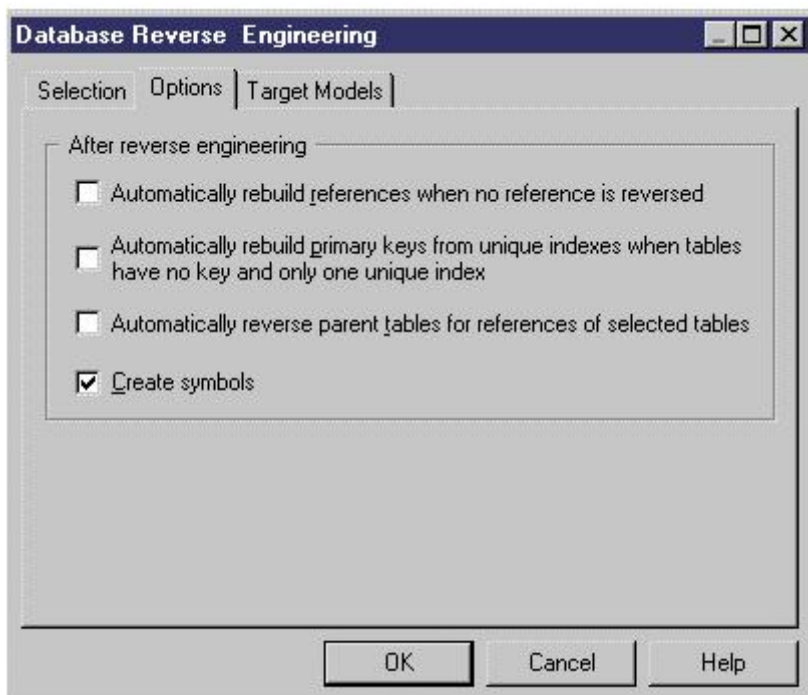


图 25

- (9) 点击确定，输出窗口的信息指出被指定的文件完全逆向工程。
- (10) 下面的逆向工程一个 ODBC 到新的 PDM 的步骤：
- (11) 选择 File->Reverse Engineer->Database 显示新的物理数据模型的对话框。
- (12) 点击 Share 单选框。
- (13) 选择下拉列表框的一个数据库管理系统。

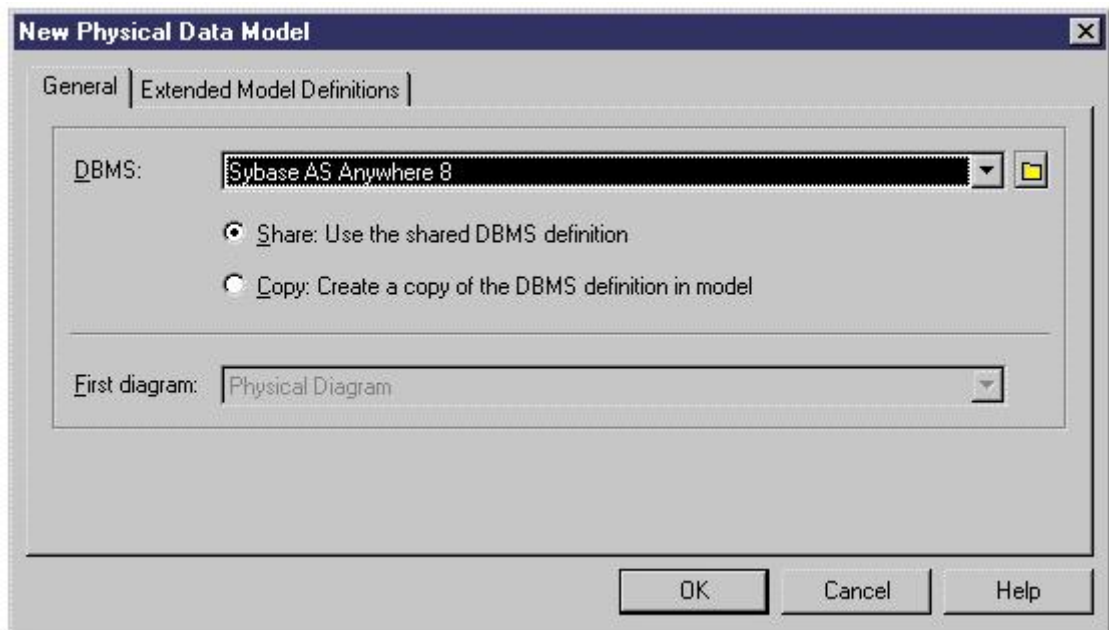


图 26

- (14) 点击确定。数据库逆向工程对话框出现。
- (15) 点击使用一个 ODBC 单选框, 选择一个 ODBC。

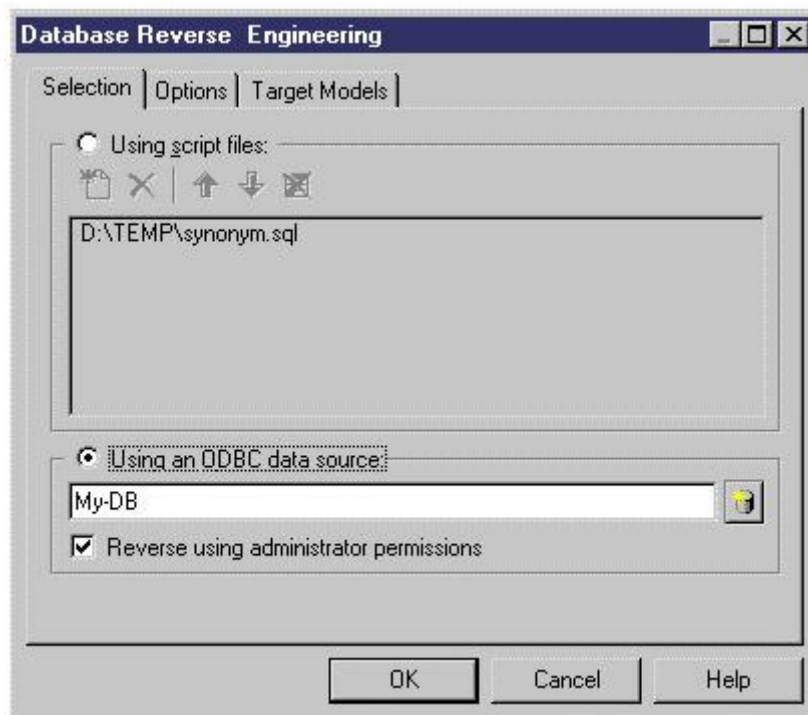


图 27

(16) 点击选项定位键显示选项页。

(17) 选择逆向工程选项。

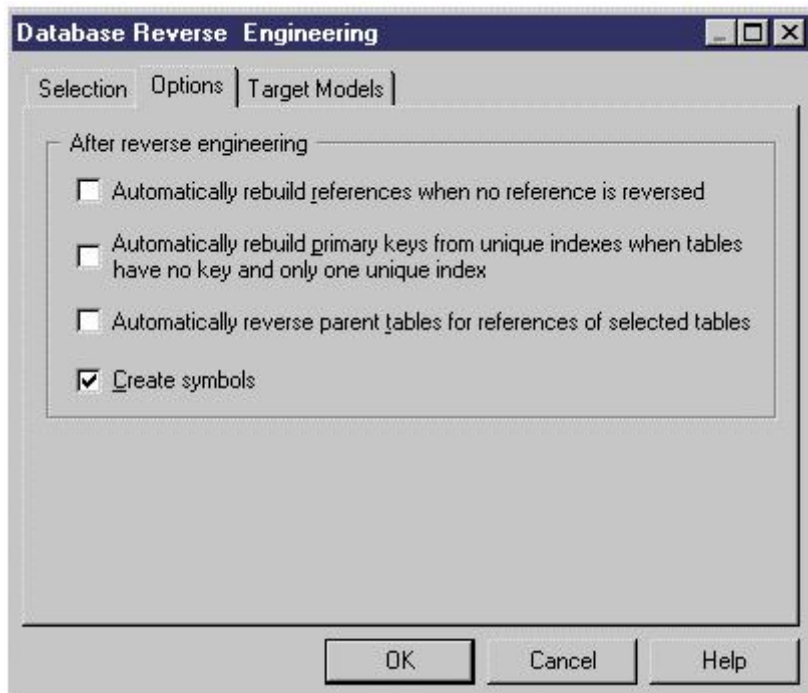


图 28

(18) 点击确定。ODBC 逆向工程对话框出现。

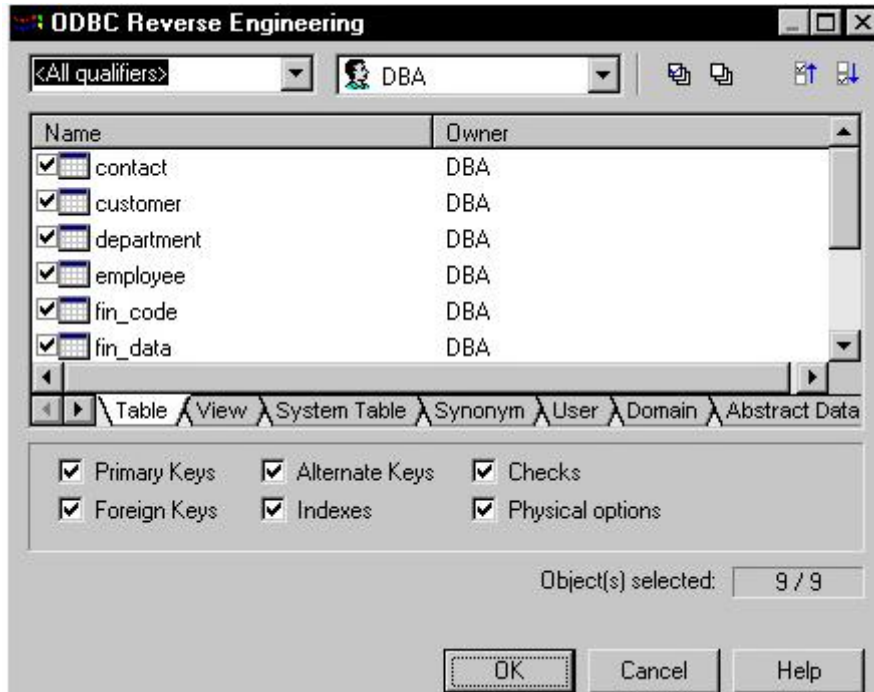


图 30

(19) 在上面部份对话框的下拉列表框中选择限定词和拥有者。

(20) 点击一个对象类型定位键。

(21) 点击确定。

输出窗口的信息显示哪些表被转换而且指出数据库成功逆向工程。

## 第九章 MS SQLSERVER 2008 简介

### 9.1 操作简介

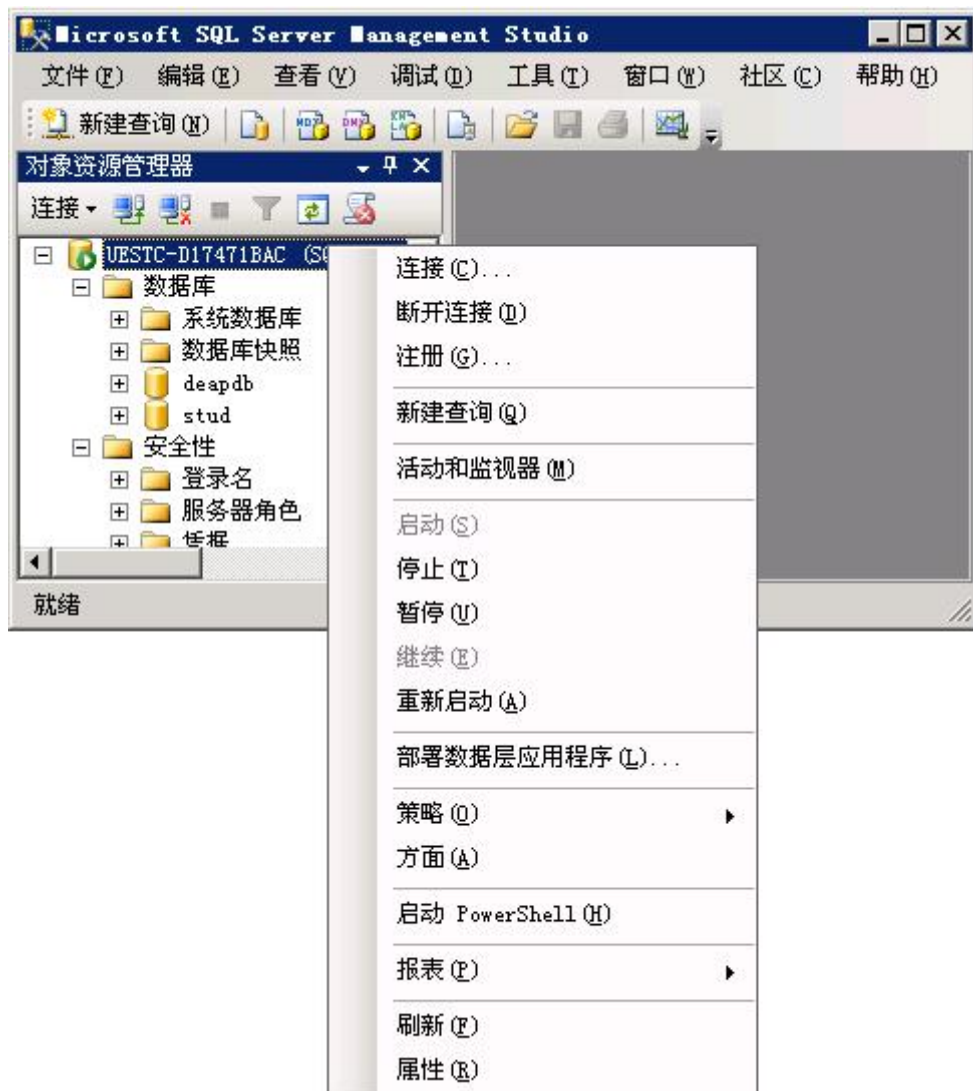
运行 SQL Server Management Studio, 服务器类型选择“数据库引擎”, 服务器名称输入服务器的 ip 地址, 如果 sqlserver 就安装在当前正在使用的这台计算机上, 可以输入“localhost”或“127.0.0.1”, 身份验证选择“windows 身份验证”。



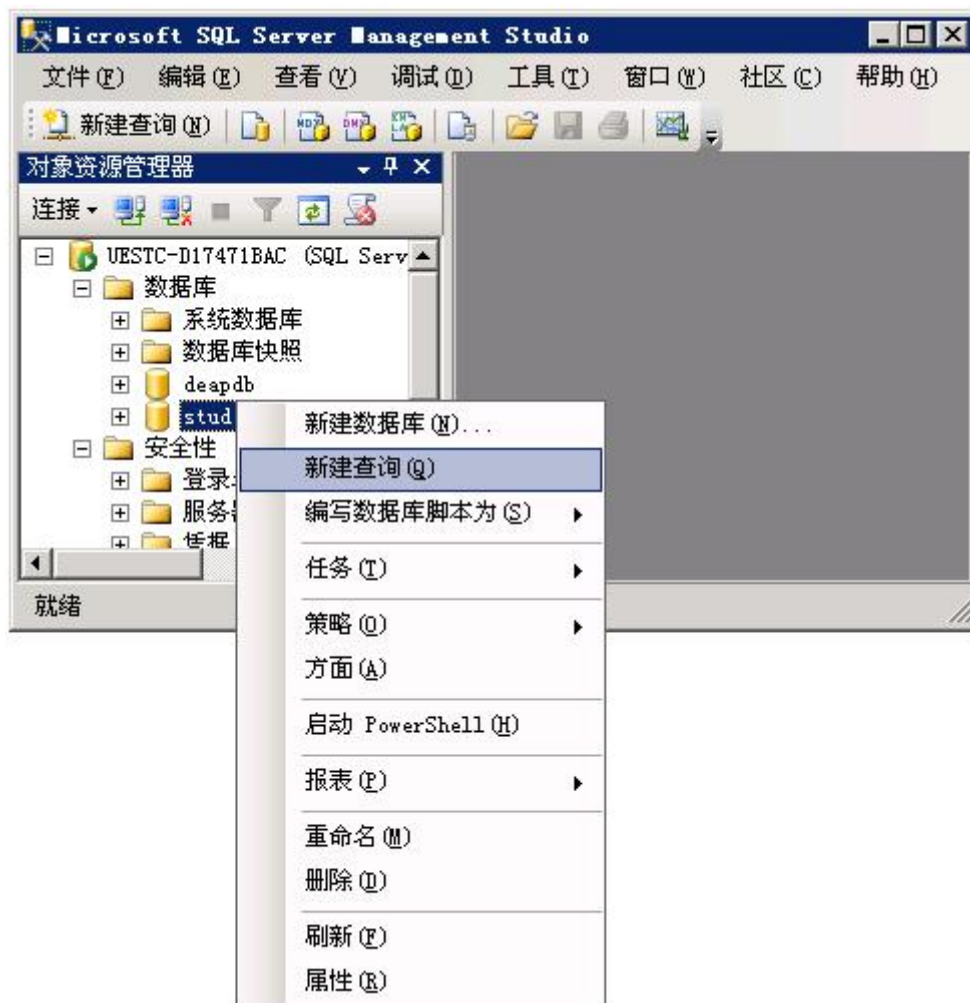
SQL Server 数据库系统提供了两种身份验证方式,“sql server 身份验证”和“windows 身份验证”,“sql server 身份验证”使用 sql server 自身的用户管理功能完成验证,管理员用户名为“sa”。

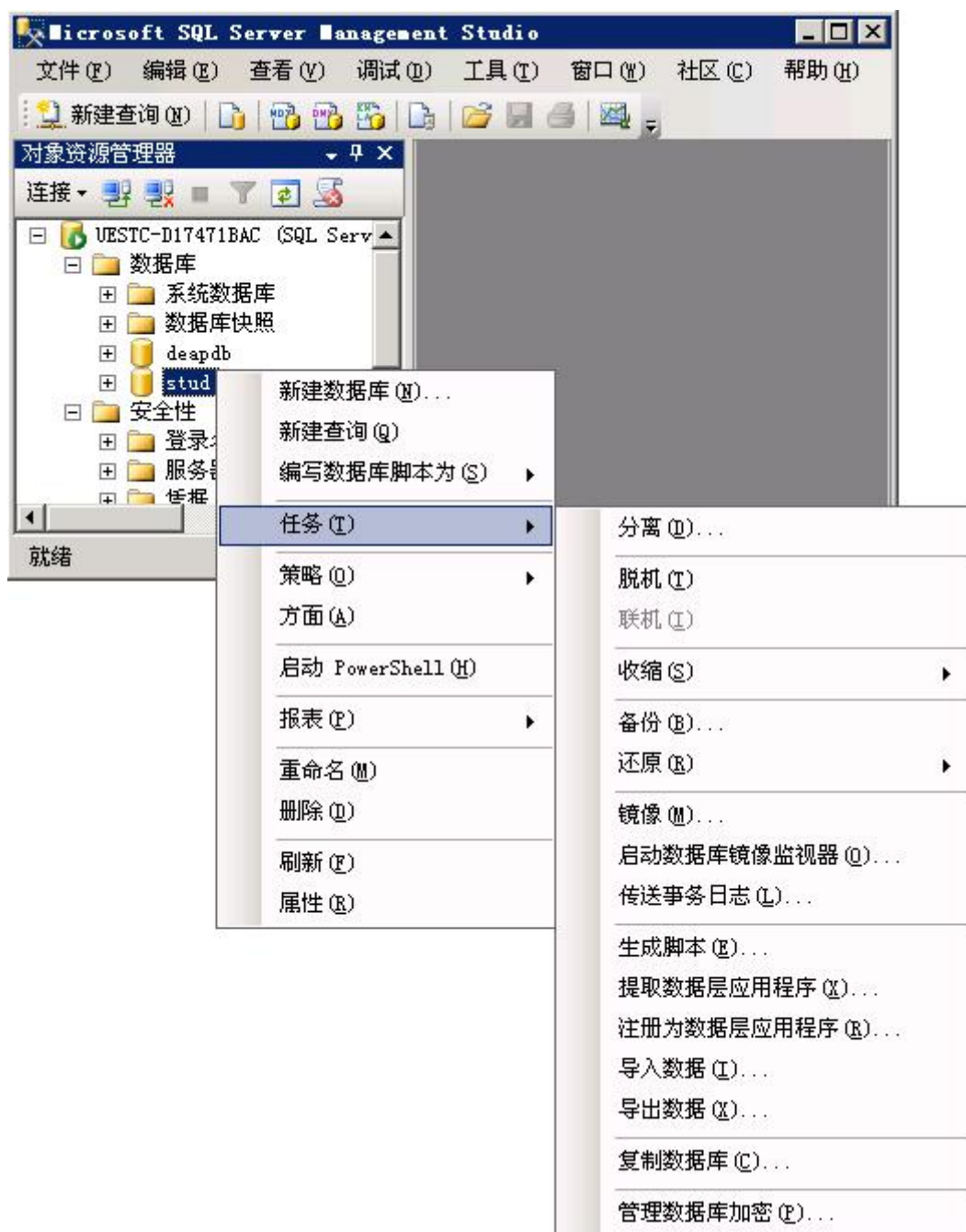
“windows 身份验证”方式使用 windows 系统的用户管理功能完成用户验证,判断当前登录到 windows 系统的用户是否有权限登录到 sqlserver 系统,对于学习者来说使用起来更方便。

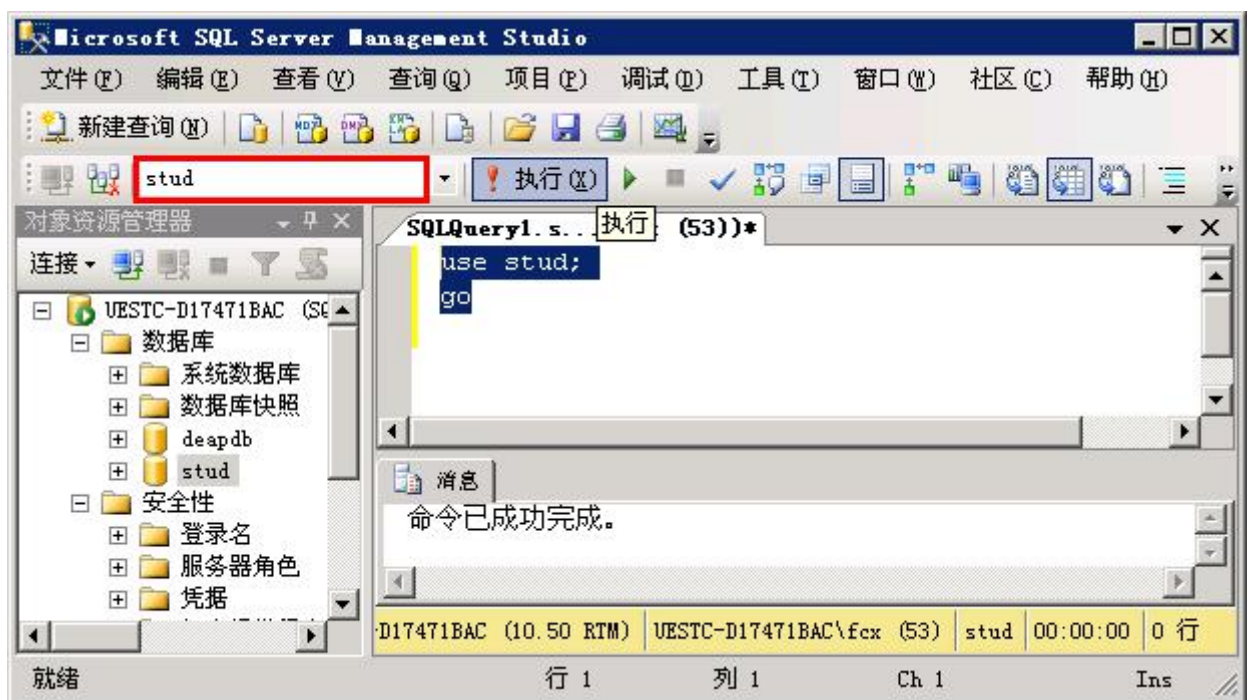












## 9.2 MS SQLSERVER 函数

### 9.2.1 统计函数（组函数）

1. AVG() :求平均值
2. COUNT() :统计数目
3. MAX() :求最大值
4. MIN() :求最小值
5. SUM() :求和

例:

--AVG 计算每个部门的平均工资

```
select avg(e_wage) as dept_avgWage  
from employee  
group by dept_id;
```

--MAX 求工资最高的员工姓名

---

```

select e_name
from employee
where e_wage = (select max(e_wage)
from employee);

```

6. --STDEV() 函数返回表达式中所有数据的标准差
7. --STDEVP() 函数返回总体标准差
8. --VAR() 函数返回表达式中所有值的统计变异数
9. --VARP() 函数返回总体变异数

### 9.2.2 数学函数

- ```

    /***三角函数***/
1. SIN(float_expression) --返回以弧度表示的角的正弦
2. COS(float_expression) --返回以弧度表示的角的余弦
3. TAN(float_expression) --返回以弧度表示的角的正切
4. COT(float_expression) --返回以弧度表示的角的余切
    /***反三角函数***/
5. ASIN(float_expression) --返回正弦是 FLOAT 值的以弧度表示的角
6. ACOS(float_expression) --返回余弦是 FLOAT 值的以弧度表示的角
7. ATAN(float_expression) --返回正切是 FLOAT 值的以弧度表示的角
8. ATAN2(float_expression1,float_expression2)
    --返回正切是 float_expression1 /float_expres-sion2 的以弧度
表示的角
9. DEGREES(numeric_expression)
    --把弧度转换为角度返回与表达式相同的数据类型可为
    --INTEGER/MONEY/REAL/FLOAT 类型
10. RADIANS(numeric_expression) --把角度转换为弧度返回与表达
式相同的数据类型可为 --INTEGER/MONEY/REAL/FLOAT 类型
11. EXP(float_expression) --返回表达式的指数值

```

- 
- 12. LOG(float\_expression) --返回表达式的自然对数值
  - 13. LOG10(float\_expression) --返回表达式的以 10 为底的对数值
  - 14. SQRT(float\_expression) --返回表达式的平方根  
/\*\*取近似值函数\*\*/
  - 15. CEILING(numeric\_expression) --返回>=表达式的最小整数返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型
  - 16. FLOOR(numeric\_expression) --返回<=表达式的最小整数返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型
  - 17. ROUND(numeric\_expression) --返回以 integer\_expression 为精度的四舍五入值返回的数据 --类型与表达式相同可为  
INTEGER/MONEY/REAL/FLOAT 类型
  - 18. ABS(numeric\_expression) --返回表达式的绝对值返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型
  - 19. SIGN(numeric\_expression) --测试参数的正负号返回 0 零值 1 正数或-1 负数返回的数据类型 --与表达式相同可为  
INTEGER/MONEY/REAL/FLOAT 类型
  - 20. PI() --返回值为 $\pi$  即 3.1415926535897936
  - 21. RAND([integer\_expression]) --用任选的  
[integer\_expression]做种子值得出 0-1 间的随机浮点数

### 9.2.3 字符串函数

- 1. ASCII() --函数返回字符表达式最左端字符的 ASCII 码值
- 2. CHAR() --函数用于将 ASCII 码转换为字符  
--如果没有输入 0 ~ 255 之间的 ASCII 码值 CHAR 函数会返回一个 NULL 值
- 3. LOWER() --函数把字符串全部转换为小写
- 4. UPPER() --函数把字符串全部转换为大写
- 5. STR() --函数把数值型数据转换为字符型数据

- 
6. LTRIM() --函数把字符串头部的空格去掉
  7. RTRIM() --函数把字符串尾部的空格去掉
  8. LEFT(), RIGHT(), SUBSTRING() --函数返回部分字符串
  9. CHARINDEX(), PATINDEX() --函数返回字符串中某个指定的子串出现的开始位置
  10. SOUNDEX() --函数返回一个四位字符码  
--SOUNDEX函数可用来查找声音相似的字符串但 SOUNDEX 函数对数字和汉字均只返回 0 值
  11. DIFFERENCE() --函数返回由 SOUNDEX 函数返回的两个字符表达式的值的差异  
--0 两个 SOUNDEX 函数返回值的第一个字符不同  
--1 两个 SOUNDEX 函数返回值的第一个字符相同  
--2 两个 SOUNDEX 函数返回值的第二二个字符相同  
--3 两个 SOUNDEX 函数返回值的第二二三个字符相同  
--4 两个 SOUNDEX 函数返回值完全相同

12. QUOTENAME() --函数返回被特定字符括起来的字符串

**例:**

```
/*select quotename('abc', '{') quotename('abc')
```

运行结果如下

```
--{  
{abc} [abc]*/
```

13. REPLICATE() --函数返回一个重复 character\_expression 指定次数的字符串

**例:**

```
/*select replicate('abc', 3) replicate('abc', -2)
```

运行结果如下

```
--- ---
```



---

```
abccabccabc NULL*/
```

14. REVERSE () --函数将指定的字符串的字符排列顺序颠倒

15. REPLACE () --函数返回被替换了指定子串的字符串

**例:**

```
/*select replace('abc123g', '123', 'def')
```

运行结果如下

```
--- ---
```

```
abcdefg*/
```

16. SPACE () --函数返回一个有指定长度的空白字符串

17. STUFF () --函数用另一子串替换字符串指定位置长度的子串

## 9.2.4 数据类型转换函数

1. CAST () 函数语法如下

```
CAST() (<expression> AS <data_ type>[ length ])
```

2. CONVERT () 函数语法如下

```
CONVERT() (<data_ type>[ length ], <expression> [, style])
```

**例:**

```
select cast(100+99 as char) convert(varchar(12), getdate())
```

运行结果如下

```
--
```

```
199 Jan 15 2000
```

## 9.2.5 时间函数

1. GetDate ( ) 返回系统目前的日期与时间

当前系统日期、时间 select getdate()

- 
2. DateAdd (interval,number,date) 以interval指定的方式, 加上number之后的日期.

例如: 向日期加上2天

```
select dateadd(day,2,'2004-10-15') --返回: 2004-10-17 00:00:00.000
```

3. datediff 返回跨两个指定日期的日期和时间边界数。

```
select datediff(day,'2004-09-01','2004-09-18') --返回: 17
```

4. DatePart (interval,date)

返回日期date中, interval指定部分所对应的整数值。

```
SELECT DATEPART(month, '2004-10-15') --返回 10
```

5. Datename

返回代表指定日期的指定日期部分的字符串

```
SELECT datename(weekday, '2004-10-15') --返回: 星期五
```

6. day(), month(), year() --可以与datepart对照一下

```
select 当前日期=convert(varchar(10),getdate(),120),当前时间
=convert(varchar(8),getdate(),114);

select datename(dw,'2004-10-15');

select 本年第多少周=datename(week,'2004-10-15'),今天是周几
=datename(weekday,'2004-10-15');
```

7. DateDiff(interval,date1,date2) 以interval 指定的方式, 返回date2 与 date1两个日期之间的差值 date2-date1

8. DateName (interval,date) 返回日期date中, interval指定部分所对应的字符串名称

表: 参数 interval的设定值

| 名称        | 简写      | 含义      | 取值          |
|-----------|---------|---------|-------------|
| Year      | Yy yyyy | 年       | 1753 ~ 9999 |
| Quarter   | Qq q    | 季       | 1 ~ 4       |
| Month     | Mm m    | 月       | 1 ~ 12      |
| DayOfYear | Dy y    | 一年中的第几日 | 1 ~ 366     |

---

|             |       |         |         |
|-------------|-------|---------|---------|
| Day         | Dd d  | 日       | 1 ~ 31  |
| Weekday     | Dw w  | 一周中的第几日 | 1 ~ 7   |
| Week        | Wk ww | 一年中的第几周 | 0 ~ 51  |
| Hour        | Hh h  | 时       | 0 ~ 23  |
| Minute      | Mi n  | 分钟      | 0 ~ 59  |
| Second      | Ss s  | 秒       | 0 ~ 59  |
| Millisecond | Ms -  | 毫秒      | 0 ~ 999 |

### 例:

`DateDiff('s','2005-07-20','2005-7-25 22:56:32')` 返回值为 514592 秒

`DateDiff('d','2005-07-20','2005-7-25 22:56:32')` 返回值为 5 天

`DatePart('w','2005-7-25 22:56:32')` 返回值为 2 即星期一(周日为1, 周六为7)

`DatePart('d','2005-7-25 22:56:32')` 返回值为 25即25号

`DatePart('y','2005-7-25 22:56:32')` 返回值为 206即这一年中第206天

`DatePart('yyyy','2005-7-25 22:56:32')` 返回值为 2005即2005年