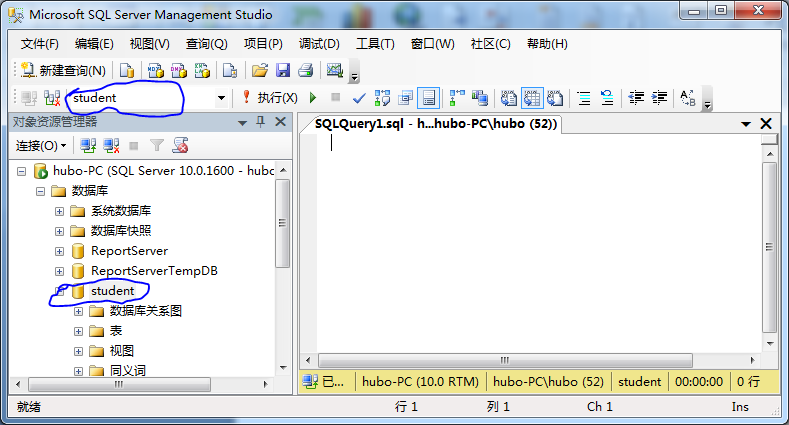
# 运行及连接

启动“配置工具”中的“SQL server配置管理器”，然后启动“SQL server(MSSQlSERVER)”，再关闭“SQL server配置管理器”,启动“Microsoft SQL Server Management Studio”.此时即可正常运行了.

# 编辑环境

1. SQL Server是对大小写不敏感的语言，即不区分大小写。
2. 在新建查询时注意：



新建的查询应该是选择对象为当前的数据库，并且新建表后需要在数据库名称上右击，选择“刷新”，此时才会显示新建的表.

# 数据定义-----创建基本表

## （1）基本格式如下：

CREATE TABLE <表名> (<列名> <数据类型> [列级完整性约束条件]

[,<列名> <数据类型> [列级完整性约束条件] ]

…………

[,<表级完整性约束条件]> ] ) ;

例如：建立一个“课程”表course

create table Course(

Cno char(4) primary key ,/\*列级完整性约束\*/

Cname char(40) ,

Cpno char(4) ,

Ccredit smallint ,

foreign key (Cpno) references Course(Cno) /\*表级完整性约束\*/

);

注：表级完整性约束条件与列级完整性约束条件均可以有多个.

## （2）完整性约束如下：

1. 主码约束： PRIMARY KEY
2. 唯一性约束：UNIQUE
3. 非主值约束：NOT NULL
4. 参照完整性约束

注：PRIMARY KEY = UNIQUE + NOT NULL

# 数据定义-----修改基本表

## （1）一般格式如下：

ALTER TABLE <表名>

[ADD < 新列名> <数据类型> [完整性约束] ]

[DROP <完整性约束名> ]

[ALTER COLUMN <列名> <数据类型> ] ;

其中：

ADD 子句用于增加新列和新的完整性约束条件；

DROP子句用于删除指定的完整性约束条件 ；

ALTER COLUMN 子句用于修改原有的列定义，包括修改列名和数据类型。

注：无论表中是否有数据，新增加的列一律为空.

## （2）新增列

向Student表增加“入学时间”列，其数据类型为日期型。

ALTER TABLE Student ADD Sentreac DATE ;

## （3）修改数据类型

将年龄的数据类型改为char(假设原来为int).

ALTER TABLE Student ALTER coulumn Sage char.

## （4）增加约束条件

增加课程名称必须惟一值的约束条件

ALTER TABLE Course ADD UNIQUE (Cname) ;

## （5）删除某一列

格式如下：

Alter table <表名> drop column <列名> ;

例如：删除COMPANY数据库PROJ表中的Begindate和Enddate两列。

alter table [COMPANY].[dbo].[PROJ] drop coLumn Begindate ;

alter table PROJ drop column Enddate ;

# 基本数据类型

|  |  |
| --- | --- |
| 数据类型 | 含义 |
| Char(n) | 长度为n的定长字符串 |
| Varchar(n) | 最大长度为n的变长字符串 |
| Int | 长整数 |
| Smallint | 短整数 |
| Numeric(p,d) | 定点数，由p位数字(不包括符号、小点数)组成，小数后面有d位数字 |
| Real | 取决于机器精度的浮点数 |
| Double precision | 取决于机器的双精度浮点数 |
| Float(n) | 浮点数，精度至少为n位数字 |
| Date | 日期，包含年、月、日，格式为YYYY-MM-DD |
| Time | 时间，包含一日的时、分、秒，格式为：HH-MM-SS |

# 数据定义-----删除基本表

## 删除基本表的一般格式

Drop table <表名> [restrict|cascade] ;

说明：

1. 若选择restrict则该表的删除是有限条件的欲删除的基本表不能被其它表的约束所引用(如check , foreign key等约束)，不能有视图，不能有触发器，不能有存储过程和函数。如果存在这些依赖对象则该表不能被删除。
2. 若选择cascade，则该表的删除没有限制条件。在删除基本表的同时，相关的依赖项，例如视图，都被一起删除。

## 举例

删除DEPT表：

drop table DEPT ;

# 数据更新-----插入数据

## (1)插入元祖

A）格式如下：

Insert

Into <表名> [(<属性列1> [,<属性列2>]……)] ;

Values (<常量1> [, <常量2>]……) ;

B）说明：

Into子句中没有出现的属性列，新元祖在这些列上将去空值；

如过某个常量为空或者不确定的话可以用 <null>取代.

C）举例：

向dept表添加14号部门客服中心，地址为开发区紫光路2号。

insert

into dept(DNO ,DNSME ,ADDR)

values (14 ,'部门客服中心' ,'开发区紫光路号') ;

向dept表添加15号部门技术支持部，地址不详。

insert

into dept

values (15 ,'部门技术支持部' ,NULL) ;

向empl表添加1401号员工张山，入职时间为2007年9月1日，工资2000元，14号部门。

insert

into EMPL

values (1401 ,'张山' ,'2007-9-1' ,2000 ,14) ;

注意：字符串要用单引号括起来，标点符号一律是英文状态下输入的

## (2)插入子查询结果

A）格式：

Insert

Into <表名> [(<属性列1> [,<属性列2>]……)]

子查询 ;

B）举例：

对每一个系，求学生的平均年龄，并把结果存入数据库

Insert

Into DEPT\_age (Sdept ,Ava\_age)

Select Sdept , AVG(Sage)

From Student

Group by Sdept ;

# 数据更新------修改数据

## 修改数据的一般格式

Update <表名>

Set <列名> = <表达式> [,<列名> = <表达式>]……

[where <条件>] ;

## 修改某一个元组的值

把dept表中客服中心的地址改为“大连甘井子区红岭路”。

update dept

set ADDR = '大连甘井子区红岭路'

where DNAME = '客服中心' ;

## 修改多个元组的值

把job表中参与104号项目的每人天数增加5天。

update JOB

set DAYS = DAYS + 5

where PNO = 104 ;

把job表中的每人天数乘以系数0.8。

update JOB

set DAYS = DAYS \* 0.8

## 带子查询的的修改语句

将计算机科学系全体学生的成绩置零

Update SC

Set Grade = 0

Where ‘CS’ =

(select Sdept

From Student

Where Student.Sno = SC.Sno) ;

# 数据更新----删除数据

## （1）一般格式

Delete

From <表名>

[where <条件>] ;

## 删除某一个元组的值

删除dept表中15号部门的信息。

delete

from dept

where DNO = 15 ;

## 删除多个元组的值

删除dept表中地址为空的部门。

delete

from dept

where ADDR is NULL ;

删除empl表中2008年入职且工资低于1500元的员工。

delete

from EMPL

where HIREDATE like '2008%' and SALARY < 1500 ;

## 带子查询的删除语句

Delete

From SC

Where ‘CS’ = ( select Sdept

From Student

Where Student.Sno = SC.Sno ) ;

# 数据传输-----运行脚本添加数据

步骤如下：(以添加到COMPANY数据库为例)

(1)使用记事本编辑dept.sql文件(可以新建一个文本文档然后将其扩展名改为.sql)，添加数据如下：

Insert into dept values(10,’市场销售部’，’大连市软体园路8号’);

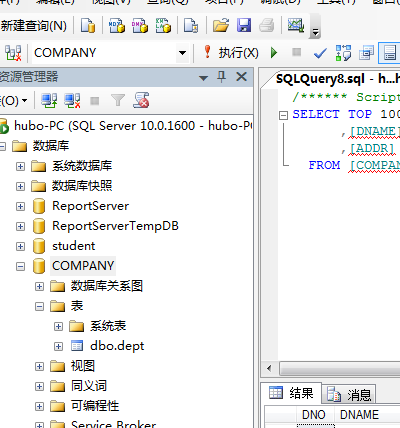
Insert into dept values(11,’财务部’，’大连市软体园路6号’);

Insert into dept values(12,’研发部’，’大连市软体园路9号’);

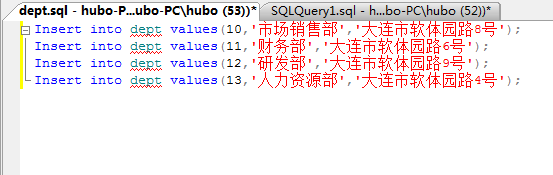
Insert into dept values(13,’人力资源部’，’大连市软体园路4号’);

关闭并保存文件。

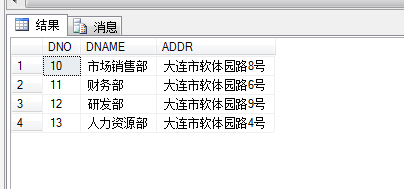
(2)选择菜单“文件” 🡪“打开”🡪“文件(F)”选择COMPANY为当前数据库，如图：



depa.sql的语句被显示在编辑窗口，按F5执行查询，如图：



执行成功后显示的结果如图：



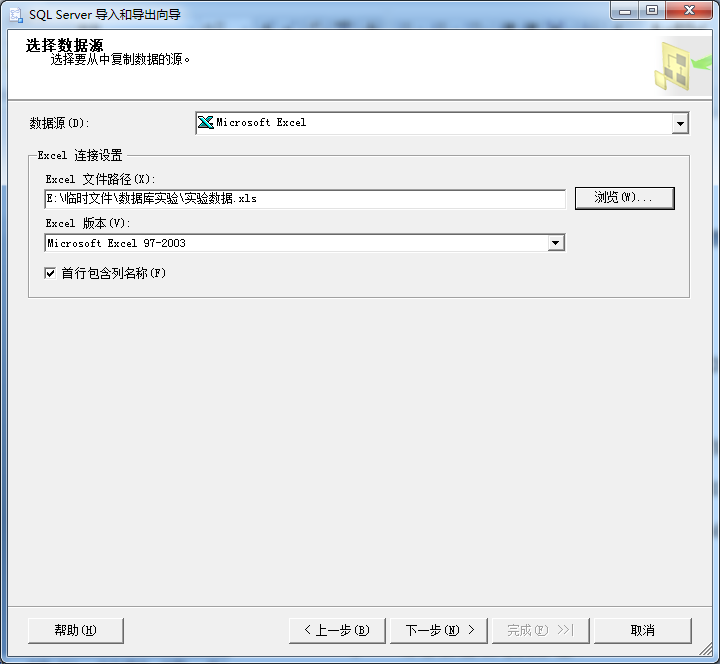
# 数据传输----从Excel表导入数据到数据库

步骤如下：(以导入到COMPANY数据库中的DEPT表为例)

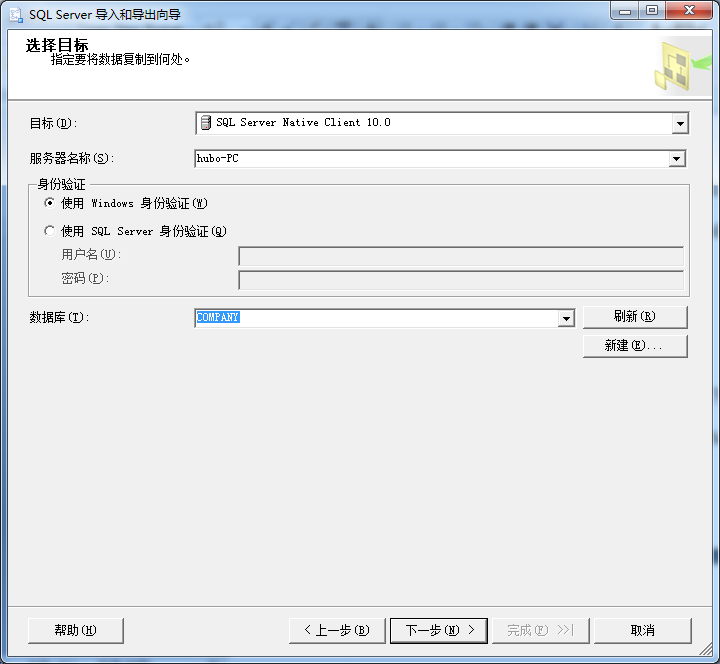
(1)开始菜单---🡪导入和导出数据，如下图：



(2) 选择数据源为Microsoft Excel 97-2000,如图：(数据源为已经建好数据的表格)



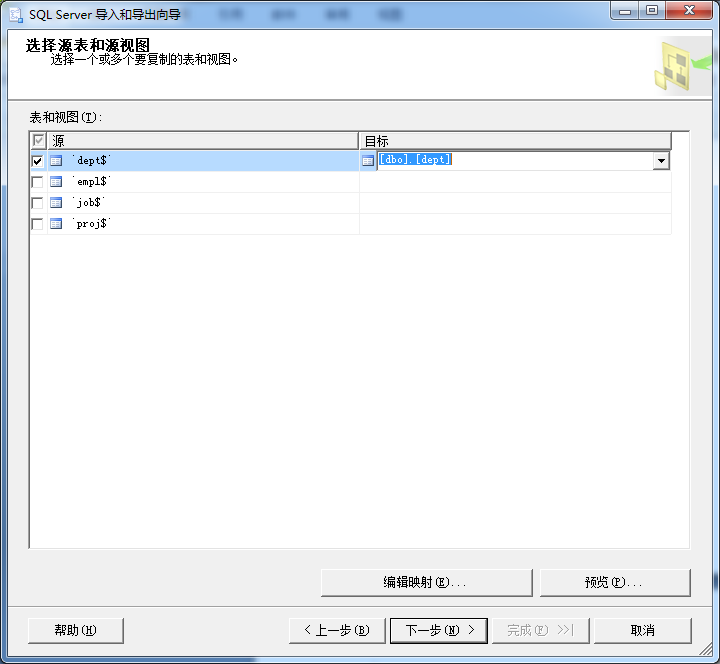
选择目的为COMPANY数据库，如图：



指定从源数据库复制表和视图，如图：



选择源表为DEPT$,目的表为[COMPANY].[dbo].[DEPT],如图：

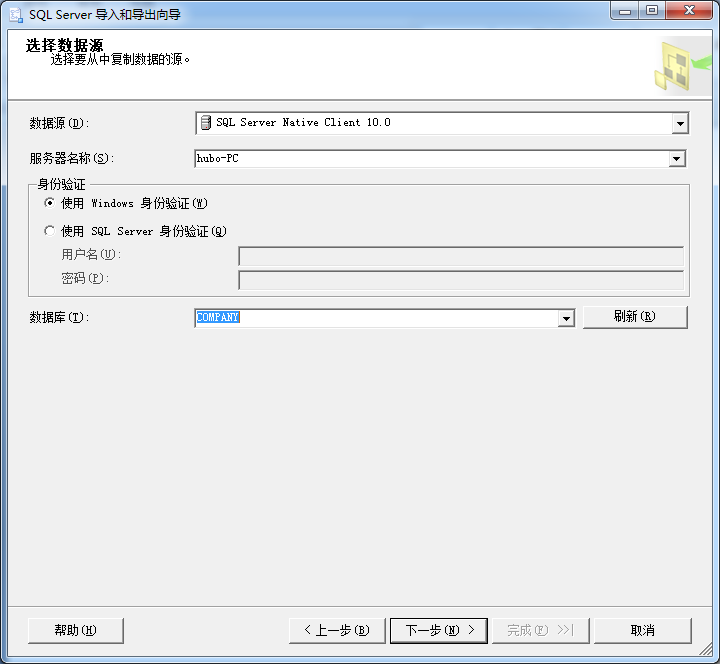


立即运行，数据即可导入到指定的表。

# 数据传输------从数据库导出数据到Excel表

步骤如下：(以从COMPANY数据库的表dept导出数据到excel表格为例)

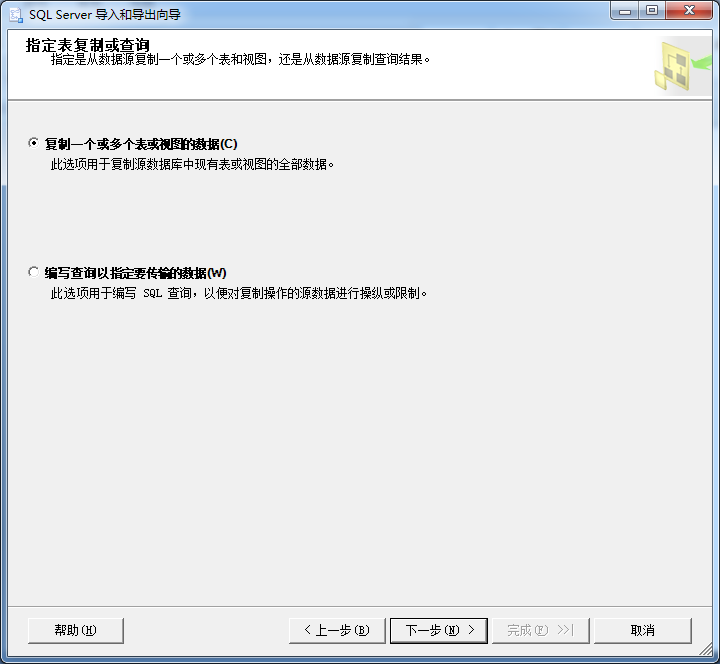
(1)启动“导入和导出数据(32)”，按照上面的导入数据的步骤操作,选择数据源为COMPANY数据库，如图：



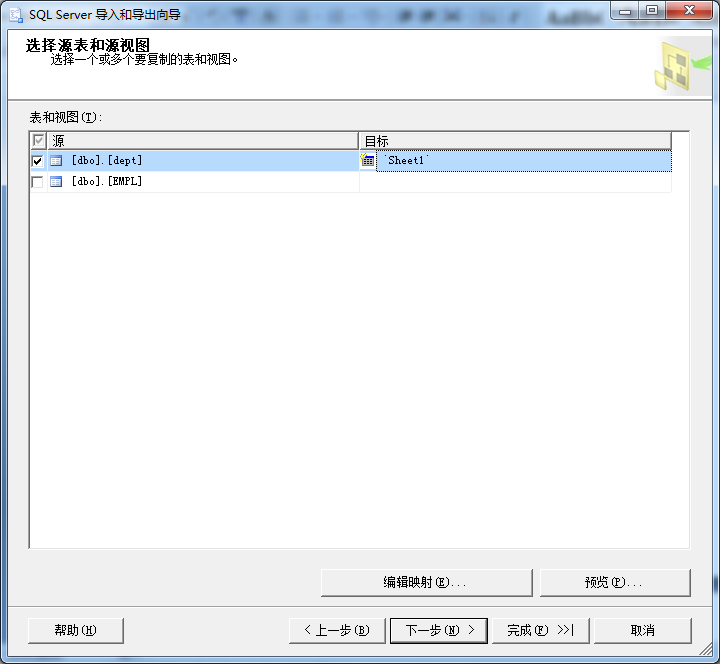
选择目的为Excel文件，并为文件命名，如图：



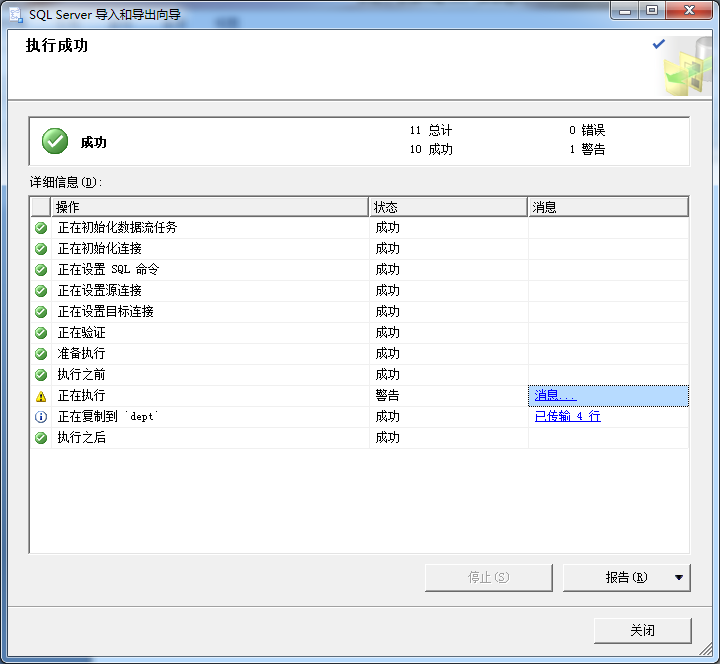
指定从源数据库复制表和视图，如图：



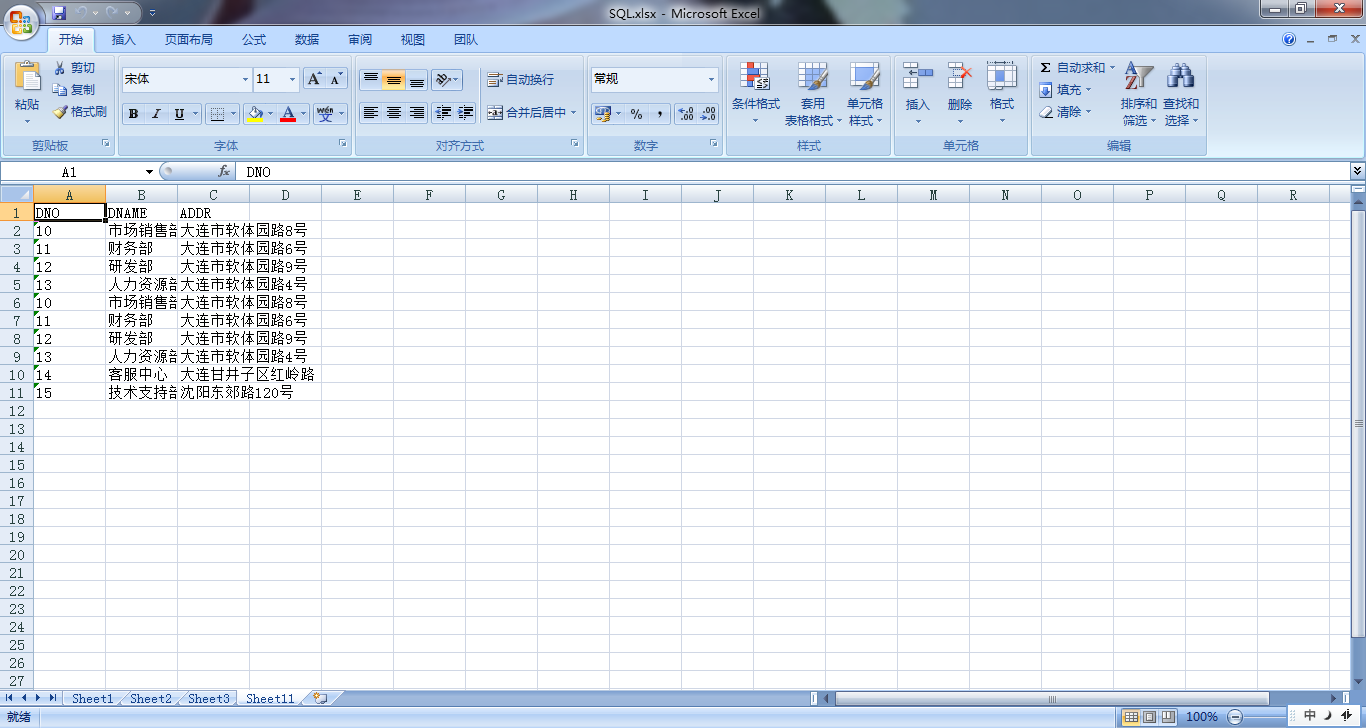
选择源表为DEPT，如图：



立即运行，单击“完成”按钮，执行成功，如图：



打开DEPT.XLS查看结果，如图：



# 数据查询-----单表查询

## (1)查询语句格式

Select [all|distinct] <目标列表达式> [,<目标列表达式>]……

From <表名或视图> [,<表名或视图>]……

[where <条件表达式>]

[group by <列名1> [having <条件表达式>]]

[order by <列名2> [asc|desc]] ;

注：[all|distinct]中all为缺省值，取消结果中的重复列则用distinct；

[asc|desc]中asc为缺省值，表示按照升序排列。对于空值，若按照升序排，则含空值的元组显示在最后面；若按降序排，则空值的元组最先显示。

## (2)查询指定列

a)查询部门表dept中所有部门的详细信息，并且列名用汉字表示。

select DNO,DNAME,ADDR

from dept ;

b)查询部门表dept中人力资源部的部门编号。

select DNO

from dept

where DNAME = '人力资源部' ;

<目标列表达式>中各个列的先后顺序可以与表中的顺序不一致.

## (3)查询全部列

查询全体学生的详细记录

Select \*

From Student ;

## (4)将查询结果的列名用别名显示

查询部门表dept中所有部门的详细信息，并且列名用汉字表示。

select DNO 部门编号 ,DNAME 部门名称 ,ADDR 部门地址

from dept ;

## (5)在查询的结果中插入新的一列用来显示指定的内容

Select Sname NAME ,’Year of Birth:’ BIRTH ,Sbirth BIRTHDAY ,Sdept DEPARTMENT

From Stuent ;

则显示的结果中，每个元组的第二列均为”Year of Birth:”，此列在原数据库中是不存在的.

## (6)查询经过计算的值

Select 子句的<目标列表达式> 不仅可以是表中的属性列，也可以是表达式。

例：查询全体学生的姓名及其出生年月

Select Sname ,2004 – Sage /\*当时年份减去年龄为出生年月

From Stufent ;

## (7)设置查询显示的字母全为大写(或小写)

Select Sname ,’Year of Birth:’ ,2004 – Sage ,LOWER(Sdept)

From Student ;

此时Sdept显示的结果全为小写

Select Sname ,’Year of Birth:’ ,2004 – Sage ,UPPER(Sdept)

From Student ;

**注：要设置查询表中的属性列名的大小写可以用LOWER,UPPER。**

## (8)消除取值重复的行

a)查询雇员表empl中出现的所有部门编号，要求无重复。

select distinct DNO

from EMPL ;

b)查询项目表proj中所有项目名称。

select PNAME

from PROJ

等价于

select all PNAME

from PROJ

## (9)查询满足条件的元组

常用的查询条件

|  |  |
| --- | --- |
| 查询条件 | 谓词 |
| 比较 | =, > , < ,>= , <= ,!= ,<> ,!> , !< ,NOT+上述比较运算符 |
| 确定范围 | Between and ,not between and |
| 确定集合 | In , not in |
| 字符匹配 | Like ,not like |
| 空值 | NULL ,NOT NULL |
| 多重条件(逻辑运算) | And ,or ,not |

注：Between后是范围的下限，and后是范围的上限.查询结果中包含上下限的结果.

1. 查询成绩不及格的学生的学号

Select distinct Sno

From SC

Where Grade < 60 ;

1. 查询年龄在20到30**(包含20及30)**之间的学生的姓名、系别和年龄

Select Sname ,Sdept ,Sage

From Student

Where Sage between 20 and 30 ;

1. 查询计算科学系(CS) ,数学系(MA) ,信息系(IS)学生的姓名和性别

Select Sname ,Ssex

From Student

Where Sdept in (‘CS’ ,’MA’ ,’IS’) ;

## (10)含通配符”%”的查询

%(百分号)：代表任意长度(长度可以为0)的字符串，例如a%b代表以a开头且以b结尾的任意长度的字符串，acb ,afdsagasdsab ,ab都满足。

例：查询雇员表empl中姓名以“伟“字结尾的员工信息。

select \*

from EMPL

where ENAME like '%伟' ;

## (11)含通配符”\_”的查询

\_(下划线)：代表任意单个字符，例a\_b代表以a开头且以b结尾的长度为3的字符串。

例：查询姓“欧阳”且全名3个汉字的学生的姓名

Select Sname

From Student

Where Sname like ‘欧阳\_\_’ ;

注：一个汉字占两个字符的位置

## (12)字符串本身含通配符的查询

使用使用escape ‘<换码字符>’。

查询以”DB\_”开头，且倒数第三个字符为i的课程的详细情况。

Select \*

From Student

Where Cname like ‘DB\\_%i\_\_’ escape ‘\’ ;

注：escape ‘\’表示”\”为换码字符，则第一个”\_”是普通的字符，后两个”\_”表示通配符。

## (13)涉及空值的查询

例：查询所有有成绩的学生的学号

Select Sno

From SC

Where Grade IS NULL ;

注：此处“IS”不能用”=”替代

## (14)带排序的查询

查询工作表job中的工作信息，结果按工作天数升序排列。

select \*

from JOB

order by DAYS asc ;

等价于

select \*

from JOB

order by DAYS asc ;

查询雇员表empl中所有员工的详细信息，结果按员工姓名降序排列。

select \*

from EMPL

order by ENAME desc ;

## (15)含聚集函数的查询

聚集函数主要有：

Count ([distinct|all] \*) //统计元组个数

Count ([distinct|all] <列名>) //统计一列中值的个数

Sum ([distinct|all] <列名>) //计算一列值的总和(此列必须是数值型)

Avg ([distinct|all] <列名>) //计算一列值的平均值(此列必须是数值型)

Max ([distinct|all] <列名>) //求一列值中的最大值

Min ([distinct|all] <列名>) //求一列值中的最小值

注：[distinct|all]缺省时为all.

**在聚集函数遇到空值时，除count(\*)外，都跳过空值而只处理非空值.**

例：查询1号课程的学生的最高分数

Select max(Grade)

From SC

Where Cno = ‘1’ ;

## (16)到group by子句的查询

a)求各个课程号及相应的选课人数

select Cno ,count(Sno)

from SC

group by Cno ;

b)查询选修了3们以以上的课程的学生的学号

select Sno

from SC

croup by Sno

having count(\*) > 3 ;

**注：where子句与having子句短语的区别在于作用对象不同。Where子句作用于基本表或视图，从中选择满足条件的元组；having子句作用与组，从中选择满足条件的组。**

# 数据查询-----连接查询

若一个查询同时设计两个以上的表，则称之为连接查询。

## 等值与非等值连接查询

格式如下：

[<表名1>.]<列名1> <比较运算符> [<表名2>.] <列名2>

其中主要的比较运算符有：= ,> ,<,>= ,<= ,!=(或<>) 等.

此外连接谓词还可以有如下的形式：

[<表名1>.] <列名1> between [<表名2>.] <列名2> and [<表名2>.] <列名2>

当连接运算符为=时，称为等值连接，否则称为非等值连接。

注：当属性列在查询的所有表中是唯一的时候则可以去掉前面的表名，否则必须加上表名。

例：

查询每个学生及其选修课程的情况.

Select Student.\* ,SC.\*

From Student ,SC

Where Student.Sno = SC.Sno ;

## 自身连接

例：查询每一门课程的间接先修课(即先修课的先修课)

分析：此时为Course表的自身连接，故要为Course表去两个别名，一个是first,一个是second。

Select first.Cno ,second.Cno

From Course first ,Course second

Where first.Cpno = second.Cno ;

## (3)外连接

在上例中，没有显示200215123和200215125两个学生的信息，原因在于他们没有选课。有时想以Student表为主体列出每个学生的基本情况及其选课情况，则需要使用外连接。

用外连接做上面的例题：

Select Student.Sno ,Sname ,Ssex ,Sage ,Sdept ,Cno ,Cgrade

From Student LEFT JOIN SC ON (Student.Sno = Sc.Sno) ;

/\*也可以用USING来去掉上面结果中的重复值：

From Student LEFT JOIN SC USING(Sno) ; \*/

注：做链接列出左边关系(如本例)中的所有元组，右外连接列出右边关系中的所有元组。

## (4)复合条件连接

在上面的例子中，where子句中只有一个条件，即连接谓词。Where也可以有多个连接条件，称为符合条件连接。

例：查询选修2号课程且成绩在90分以上的所有学生

Select Student.Sno ,Sname

From Student ,SC

Where Student.Sno = SC.Sno AND /\*连接谓词\*/

SC.Cno = ‘2’ AND SC.Grade > 90 ; /\*其它限制条件\*/

例：

查询每个学生的学号、姓名、选修课程名及成绩

Select Student.Sno ,Sname ,Cname ,Grade

From Student ,SC ,Course

Where Student.Sno = SC.Sno and SC.Cno = Course.Cno ;

# 数据查询-----集合查询

集合查询操作主要包括并操作UNION、交操作INTERSECT和差操作EXCEPT。

注：多个集合操作的个查询结果的列数必须相同，对应项的数据类型也必须相同。

## 并操作UNION

使用UNION将多个查询结果并起来时，系统会自动的去掉重复元组；如果要保留重复元组，则可以使用UNION ALL操作符。

例：

查询计算机科学系的学生 及年龄不大于19岁的学生.

Select \*

From Student

Where Sdept = ‘CS’

Union

Select \*

From Student

Where Sage <= 19 ;

例：

查询选修了1号课程或者2号课程的学生

Select Sno

From SC

Where Cno = ‘1’

Union

Select Sno

From SC

Where Cno = ‘2’ ;

## (2)交操作(INTERSECT)

例：

查询计算机科学系的学生与年龄不大于19岁的学生的交集

Select \*

From Student

Where Sdept = ‘CS’

Intersect

Select \*

From Student

Where Sage <= 19 ;

等价于

Select \*

From Student

Where Sdept = ‘CS’ and Sage <= 19 ;

## (3)差操作(EXCEPT)

例：

查询计算机科学系的学生与年龄不大于19岁的学生的差集

Select \*

From Student

Where Sdept = ‘CS’

Except

Select \*

From Student

Where Sage <= 19 ;

# 数据查询-----嵌套查询

一个select-from-where语句称为一个查询块。

**注：子查询的select语句中不能使用order by子句，order by子句只能对最终的查询结果排序**。

## (1)带IN谓词的子查询

例：

查询与“刘晨”在同一个系学习的学生

Select Sno ,Sname ,Sdept

From Student

Where Sdept in

(select Sdept

From Student

Where Sname = ‘刘晨’) ;

## 带有比较运算符的子查询

例：

找出每个学生超过他选修课平均成绩的课程号

Select Sno ,Cno

From SC x

Where Grade >= (select AVG(Grade) /\*某个学生的平均成绩\*/

From SC y

Where y.Sno = x.Sno) ;

## 带有ANY(SOME)或ALL谓词的子查询

子查询返回单值可以用比较运算符，但是但会多值要用NAY或ALL谓词修饰。而使用ANY或ALL谓词时则必须同时使用比较运算符。

例：

查询其他系中 比计算机科学系某一学生年龄小的学生的姓名及年龄

Select Sname ,Sage

From Student

Where Sage < any (select Sage

From Student

Where Sdept = ‘CS’)

And Sdept <> ‘CS’ ;

例：

查询其他系中 比计算机科学系所有学生年龄都小的学生的姓名及年龄

Select Sname ,Sage

From Student

Where Sage < all (select Sage

From Student

Where Sdept = ‘CS’)

And Sdept <> ‘CS’ ;

## 带有EXITS或NOT EXITS谓词的子查询

例：

查询所有选修了1号课程的学生的姓名

Select Sname

From Student

Where not exits

(select \*

From Student

Where Sno = Student.Sno and Cno = ‘1’) ;

例：

查询选修了全部课程的学生的姓名

Select Sname

From Student

Where not exits

(select \*

From Course

Where not exits(select \*

From SC

Where Sno = Student.Sno and

Cno = Course.Cno)) ;

例：

查询了至少选修了学生200215122选修的全部课程的学生的号码

Select distinct Sno

From SC SCX

Where not exits(select \*

From SC SCY

Where SCY.Sno = ‘200215122’ and

not exits(select \*

from SC SCZ

where SCZ.Sno = SCX.Sno and

SCZ.Cno = SCY.Cno)) ;

# 视图

## 1、基本介绍

1. 视图是从一个或者几个基本表导出的表，它是一个虚表。**数据库中只是存放视图的定义，而不存放视图对应的数据**;
2. RDBMS在执行CREATE VIEW语句的结果只是把视图的定义存入数据字典，并不执行其中的SELECT语句。只在对视图进行查询时，才按视图的定义从表中将数据查出;
3. 视图不仅可以建立在一个或者多个表上，还可以建立在一个或者已定义好的视图上，或者建立在基本表与视图上;
4. 对视图的更新，最终要转换为对基本表的更新;
5. 视图属于数据库外模式范畴.

## 2、建立视图

格式如下：

Create view <视图名> [(<列名> [,<列名>]……)]

As <子查询>

[with check option] ;

注：子查询通常不允许含有order by 子句和distinct子句。

例如：建立信息系学生的视图

Create view IS\_Student

as

select Sno ,Sname ,Sage

from Student

where Sdept = ‘IS’ ;

例如：建立信息系学生的视图，并要求进行修改和插入操作时任需要保证视图只有信息系的学生。

Create view IS\_Student

as

select Sno ,Sname ,Sage

from Student

where Sdept = ‘IS’ ;

with check option.

例：建立信息系选修了1号课程的学生的视图。

Create view IS\_S1(Sno ,Sname ,Grade)

As

Select from Student.Sno ,Sname ,Grade

From Student ,SC

Where Sdept = ‘IS’ and

Studetn.Sno = SC.Sno and

SC.Cno = ‘1’ ;

## 3、删除视图

格式：

Drop view <视图名> [cascade] ;

例：删除视图BT\_S

Drop view BT\_S ;

## 4、查询视图

例：查询选修了1号课程的信息系的学生

Select IS\_Student.Sno ,Sname

From IS\_Student ,SC

Where IS\_Student.Sno = SC.Sno and

SC.Cno = ‘1’ ;

例：在S\_G视图中查询平均成绩在90分以上的学生学号和平均成绩

Select \*

From S\_G

Where Gave >= 90 ;

## 5、更新视图

例：将信息系学生视图IS\_Student 中学号为200215122的学生的姓名改为“刘辰”。

Update IS\_Student

Set Sname = ‘刘辰’

Where Sno = ‘200215122’ ;

例：删除信息系学生视图IS\_Student中学号为200215129的记录

Delete

From IS\_Student

Where Sno = ‘200215129’ ;

## 6、视图的作用

1. 视图能够简化用户的操作
2. 视图使用户能以多种角度看待同一数据
3. 视图对重构数据库提供了一定程度的逻辑独立性
4. 视图能够对机密数据提供安全保护
5. 适当的利用视图可以更清晰的表达查询

# 索引的建立与删除

## 1、建立索引

格式：

Create [unique] [cluster] index <索引名>

On <表名> (<列名> [,<次序> [<次序>]]……) ;

## 2、说明

1. 索引可以建立在表的以列或者多列上，各列名之间用逗号隔开，每个列名后面还可以用<次序>指定索引值的排列次序，可选asc(升序)或desc(降序)，缺省为asc.
2. Unique表名此索引的每一个索引值只对应唯一的数据记录;

例：

Create unique index Stusno on Student(Sno) ;

Create unique index Scno on SC(Sno ASC ,Cno DESC) ;

Create cluster index Stusname on Student(Sname) ;

1. 建立索引是为了减少查询操作的时间;
2. 索引是关系数据库的内部实现技术，属于内模式的范畴.

## 3、删除索引

格式如下：

Drop index <索引名> ;

例：删除Student表的Stusname索引

Drop index Stusname ;

# 授权与回收

## GRANT(授权)

一般格式：

Grant <权限> [,<权限>]……

On <对象类型> <对象名> [,<对象类型> <对象名>]……

To <用户> [,<用户>]……

[with grant option] ;

说明：

1. 发出给grant语句的可以是DBA，也可以是该数据库对象创建者，也可以是已经拥有该权限的用户。
2. 接受权限的用户可以是一个或多个用户，也可以是PUBLIC，即全体用户。
3. 如果指定了with grant option子句，则获得某种权限的用户还可以把这种权限再授予其他的用户；如果没有指定grant with option子句，则获得某种权限的用户只能使用该权限，不能传播该权限。
4. SQL允许具有with grant option的用户把相应权限或其子集传递授予其他的用户，但不允许循环授权，即被授权者不能再将权限授回给授权者或其祖先。

【例1】将查询Student表的权限授给用户U1

Grant select

On table Student

To U1

注：在SQL2008中，应将对象名去掉，即去掉此处的table.

【例2】把对Student表和Course表的全部操作权限授给用户U2和U3

Grant all privileces

On table Student ,Course

To U2 ,U3 ;

注：在SQL2008中一次只能授权一张表

【例3】把对表SC的查询权限授给所有用户

Grant select

On table SC

To public ;

【例4】把查询student表和修改学生学号的权限授给用户U4

Grant update(Sno) ,select

On table Student

To U4 ;

注：

1. 在SQL2008中，本例中的Sno不能是主码.
2. 对属性列的授权时必须明确指出属性列名

【例5】把对表SC的INSERT权限授给U5用户，并允许将此权限再授予其他用户.

Grant insert

On table SC

To U5

With grant option ;

注：此时用户U5不仅拥有了对SC表的INSERT权限，还可以传播此权限.

【例6】U5将此权限传给用户U6

Grant insert

On table SC

To U6

With grant option ;

【例7】U6还可以将此权限传给U7

Grant insert

On table SC

To U7 ;

注：u6未给U7传播的权限，因此U7不能再传播此权限.

## REVOKE(收回权限)

一般格式如下：

Revoke <权限> [,<权限>]……

On <对象类型> <对象名> [,<对象类型> <对象名>]……

From <用户> [,<用户>]……

[cascade|restrict] ;

【例8】把用户U4修改学学号的权限收回

Revoke upsate(Sno)

On table Student

From U4 ;

【例9】收回所有用户对表SC的查询权限

Revoke select

On table SC

From public ;

【例10】把用户U5对SC表的INSERT权限收回

Revoke insert

On table SC

From U5 cascade ;

注：

将用户U5的权限收回必须级联(cascade)收回，否则系统拒绝此命令的执行，因为在【例6】中，U5将对SC表的INSERT权限授给了U6，而在【例7】中U6又将此权限授给了U7。

说明：

***这里缺省值为RESTRICT，如果U6或U7还从其他用户获得对SC表的INSERT权限，则他们任然具有此权限，系统只收回直接或间接从U5获得的权限。***

## 创建数据库模式的权限

一般格式如下：

Create user <username>

[with] [DBA|RESOURCE|CONNECT] ;

## 4、权限与可执行的操作对照表

权限与可执行的操作对照表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 拥有的权限 | 可执行的操作 | | | |
| CREATE USER | CREATE SCHEME | CREATE TABLE | 登录数据库  执行数据查询和操纵 |
| DBA | 可以 | 可以 | 可以 | 可以 |
| RESOURCE | 不可以 | 不可以 | 可以 | 可以 |
| CONNECT | 不可以 | 不可以 | 不可以 | 可以，但是必须拥有相应的权限 |

# 数据库角色

## 角色的创建

一般格式：

Create role <角色名>

刚刚创建的角色是空的，没有任何内容

## 角色授权

Grant <权限> [,<权限>]……

On <对象类型> <对象名>

To <角色> [,<绝色>]……

DBA和用户可以利用grant语句将权限授给某一个或几个角色.

## 3、将一个角色授予其他的角色或用户

Grant <角色1> [,<角色2>]……

To <角色3> [,<用户1>]……

[with grant option] ;

## 角色权限的收回

Revoke <权限> [,<权限>]……

On <对象类型> <对象名>

From <角色> [,<角色>]……

## 实例：通过角色实现将一组权限授予一个用户

步骤：

1)首先创建一个角色R1

Create role R1 ;

2)然后使用grant 语句，使角色R1拥有Student表的select,update,insert权限

Grrant select ,update ,insert

On table Student

To R1 ;

3)将这个角色授予王平，张明，赵林。使他们拥有角色R1包含的所有的权限.

Grant R1

To 王平,张明,赵林 ;

4)也可以通过R1来收回王平的这3个权限

Revoke R1

From 王平 ;

## 6、其他实例

【例1】角色的权限的修改

Grant delete

On table Student

To R1 ;

此时R1增加了对表Student的dalate的权限

【例2】使R1减少select权限

Revoke select

On table Student

From R1 ;

# 创建用户

## 1、一般格式

Create user <用户名>

[with] [DBA|RESOURCE|CONNECT] ;

## 2、说明

1. 缺省情况该用户具有connect权限;
2. 只有系统超级用户才能创建一个新的数据库用户

## 权限说明

权限与可执行操作对照表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 拥有的权限 | 可否执行的操作 | | | |
| CREATE USER | CREATE SCHEME | CREATE TABLE | 登录数据库、执行数据查询和操纵 |
| DBA | 可以 | 可以 | 可以 | 可以 |
| RESOURCE | 不可以 | 不可以 | 可以 | 可以 |
| CONNECT | 不可以 | 不可以 | 不可以 | 可以，但必须拥有相应的权限 |

# 模式的定义与删除

## 模式与表

每一个基本表都属于一个模式，一个模式包含多个基本表

## 定义基本表时定义其所属的模式方法

【方法一】：

在表名中明显地给出模式名：

CREATE SCHEMA “S\_T”.STUDENT(……) ;

【方法二】：

在创建模式时同时创建表：

CREATE SCHEMA “S\_T” AUTHORIZATON WANG ;

CREATE TABLE TABLE1 (

COL1 SMALLINT ,

COL2 CHAR(20) ,

COL3 INT ,

COL4 NUMERIC(10,3) ,

COL5 DECIMAL(5,2)

) ;

【方法三】：

设置所属的模式：

SET SEARCH\_PATH TO “S\_T” ,PUBLIC ;

注：搜索路径的当前默认值为:$user ,PUBLIC.

## 创建模式

创建模式一般格式:

CREATE SCHEMA <模式名> AUTHORIZATION <用户名> ;

如果没有指定模式名，则<模式名>隐含为用户名.

例：定义一个学生---课程模式S—T

create schema "S\_T" authorization s1 ;

## 删除模式

删除模式的一般定义如下：

DROP SCHEMA <模式名> [CASCADE|RESTRICT] ;

# 规范化理论

## 规范化目的

1. 消除插入、删除和更新异常；
2. 解决数据冗余问题

## 第一范式

第一范式(1 NF)：每个分量都是不可分的数据项

满足第一范式是一张二维表最起码的要求

## 3、函数依赖

1)**平凡函数依赖**：，但，则称是平凡的函数依赖.

2)非平凡函数依赖：，但，则称是非平凡的函数依赖.

3)完全函数依赖：在R(U)中，如果，并且对于X的任何一个真子集，都有，则称Y对X完全函数依赖，记作：.

4)部分函数依赖：若，但Y不完全依赖于X，则称Y对X部分函数依赖，记作：

.

## 4、码

1. 主属性：包含在任何一个候选码中的属性
2. 非主属性(非码属性)：不包含在任何一个候选码中的属性
3. 全码：整个属性组都是码
4. 外码：关系模式R中属性或属性组X并非R的码，但X是另外一个关系模式的码，则称X是R的外部码(也称外码).

## 范式

1）各种范式之间的关系：



2）各种范式之间的转换



3）其它内容

一个模式中的关系模式如果都属于BCNF，那么在函数依赖的范畴内，它已实现了彻底的分离;

函数依赖是多值依赖的一种特殊情况.

1. 多值依赖与函数依赖区别

a)多值依赖的有效性与属性集的范围有关;

b)若函数在R(U)上成立，则对于任何均有成立。而多值依赖若在R(U)上成立，却不能断言对于任何有成立.

# 数据库系统概述

## 数据

数据：描述事物的符号记录.

数据是数据库中存储的基本对象.

数据与其语义是不可分的.

## 2、数据库(DS)

数据库是长期存储在计算机内部的、有组织的、可共享的数据的集合.

数据库中的数据具有永久存储、有组织、可共享的特点.

## 3、数据库管理系统(DBMS)

数据库管理系统是介于用户和计算机系统之间的一层数据管理软件.

数据库系统功能：

1. 数据定义;
2. 数据组织、存储和管理;
3. 数据操纵功能;
4. 数据的事物管理和运行管理;
5. 数据库的建立和维护功能;
6. 其它功能.

数据库管理系统的控制功能：

1. 安全性保护;
2. 完整性检查;
3. 并发控制;
4. 数据库恢复.

## 4、数据结构

数据结构是对描述的对象的类型的集合.

数据结构是对数据库系统静态特征的描述.

## 5、数据操作

数据操作是对数据库中的对象的实例允许进行的操作的集合.

数据操作是对数据库与系统动态特征的描述.

## 数据库系统

数据库系统主要包括：

1. 数据库(DB) ;
2. 数据库管理系统(DBMS) ;
3. 数据库管理员(DBA) ;
4. 应用系统.

数据库系统特点：

1. 数据结构化;
2. 数据具有较高独立性;
3. 数据共享性高，冗余度低，易扩展;
4. 有DBMS统一管理和控制.

## 数据独立性

数据独立性包括逻辑独立性和物理独立性.

逻辑独立性：用户应用程序与数据库逻辑结构是相互独立的.

物理独立性：用户应用程序与存储在磁盘上的数据库中的数据是相互独立的.

数据库系统的二级映射功能保证了数据库的独立性.

## 8、数据库三个阶段的比较

人工管理阶段特点：

1. 数据不保存;
2. 应用程序管理数据;
3. 数据不共享;
4. 数据不具有独立性.

文件系统阶段：

1. 数据可以长期存储;
2. 有文件系统管理数据;
3. 数据共享性差，冗余度大;
4. 数据独立性差.

文件系统阶段已经有软件来管理数据，并且数据可以长期存储，数据具有共享性.

应用背景方面：

1. 人工管理阶段主要应用背景是科学计算;
2. 文件系统阶段主要应用背景是科学计算与小规模数据管理;
3. 数据库系统阶段主要引用背景是大规模数据管理.

## 9、数据库系统的三级模式

三级模式：

1. 外模式(子模式);
2. 模式(逻辑模式) ;
3. 内模式(存储模式).

**注：一个数据库只有一个模式，只有一个内模式，可以有多个外模式。但是一个应用程序只能有一个外模式.**

# 数据模型

## 1、定义

数据模型是一组严格定义的概念的集合.

## 2、功能

数据模型是数据库系统的核心和基础.

## 3、三要素

数据模型三要素：

1. 数据结构;
2. 数据操作;
3. 完整性约束.

数据结构是刻画数据模型性质最重要的方面.

## 4、数据模型分类

数据模型可以分为两类：

1. 概念模型：用于数据库的设计
2. 逻辑模型和物理模型：用于数据库的实现.

# 概念模型

## 1、实体

实体是可观存在的且可相互区别的事物.

## 2、属性

属性是实体的某一特性

## 3、码

码是可以惟一标识实体的属性集

## 4、域

属性的取值范围.

## 5、实体型

用实体名及其属性名集合来抽象和刻画同类实体.

## 6、实体集

同类实体的集合

## 7、联系

联系：

1. 内部联系：属性之间的联系
2. 外部联系：实体集之间的联系

## 8、概念模式的表示方法

常用实体—联系方法(ER模型)来表示概念模式.

在E-R模型中：

矩形表示实体型，椭圆表示属性，菱形表示联系.

# 常用数据模型

## 1、层次模型

定义：

1. 有且只有一个结点没有双亲节点，该结点称为根节点;
2. 除根节点外，其它结点有且仅有一个双亲结点.

常用实现方法：

1. 邻接法;
2. 链接法

## 2、网状模型

定义：

1. 允许一个以上的结点没有双亲结点;
2. 一个结点可以有多个双亲结点;

## 3、关系模型

关系模型是目前最重要的一种数据模型.关系模型采用关系模型作为数据的组织方式.

关系模式的完整性约束：

1. 实体完整性;
2. 参照完整性;
3. 用户自定义完整性.

优点：

1. 结构单一;
2. 建立在严格的数学基础上;
3. 有较高的数据独立性和较好的安全保密性，简化了数据库设计者的工作和数据库开发建立的工作.

**缺点：**

查询效率往往不如非关系数据模型.

# 关系数据库

## 1、关系

关系模型的数据库非常简单，值包含单一的数据结构-----关系.

关系模型中数据的逻辑结构是一张扁平的二维表.

## 2、关系类型

关系类型有三种：

1. 基本关系(基本表);
2. 查询表：查询结果对应的表
3. 视图表：虚表.

## 3、关系性质

关系具有6条性质：

1. 列是同性质的，即每一列中的分量是相同的类型的数据，来自同一个域;
2. 不同的列可以来自同一个域，不同的属性要有不同的属性名;
3. 列的顺序是可以交换的;
4. 行的顺序是可以交换的;
5. 任意两个元组的候选码不能相同;
6. 分量都是不可分的数据项.

## 4、关系模型

关系的描述称为关系模型.

关系模型三要素：

1. 关系数据结构;
2. 关系操作;
3. 关系完整性约束.

## 5、关系操作特点

关系操作的特点是集合操作，即操作的对象和结果都是关系.

## 关系代数运算

关系代数元算的对象和结果都是关系.

运算三要素：

1. 运算对象;
2. 运算符;
3. 运算结果.

## 关系基本运算

关系基本运算有五种：

1)选择();

2)投影() ;

3)笛卡尔积(×);

4)并(U) ;

5)差(-).