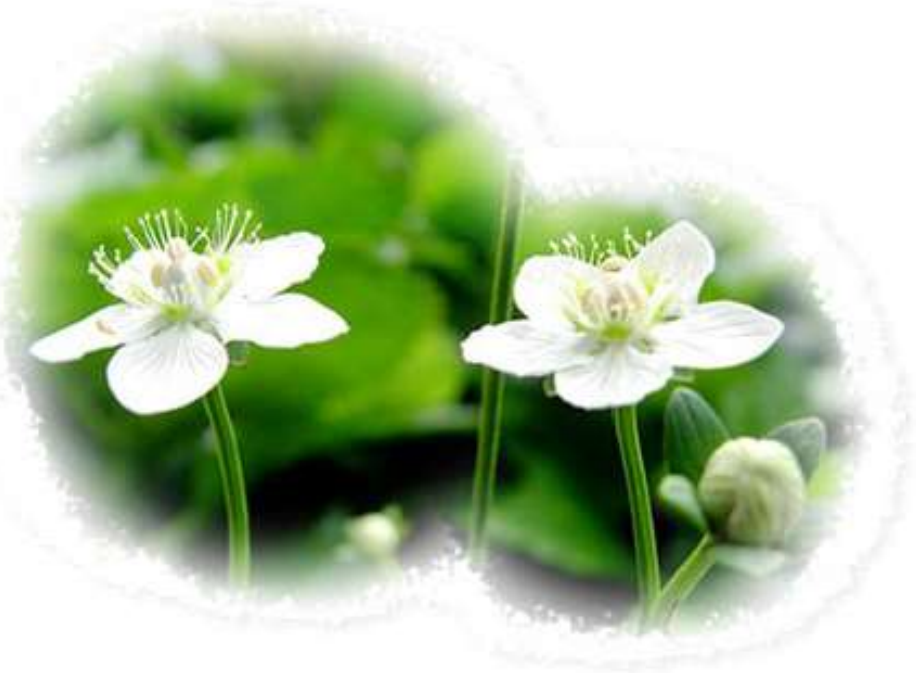
数据库系统原理

第3章　关系数据库语言3



分级通关平台，交流

基础篇

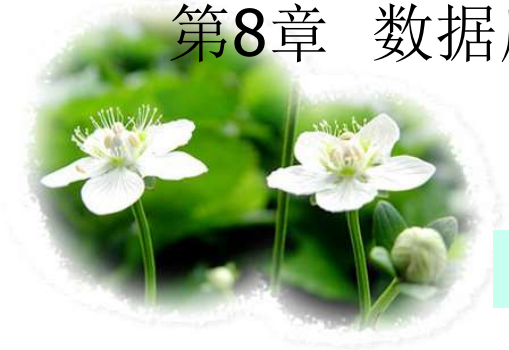
第1章 绪论
第2章 关系数据库*3
第3章 标准语言SQL*3
第4章 数据库安全性
第5章 数据库完整性
实验1

设计与应用开发篇

第6章 关系数据理论*2
第7章 数据库设计
实验2
第8章 数据库编程

系统篇

第9章 *关系查询处理和优化
实验3
第10章 数据库恢复技术
第11章 并发控制
实验4



内容提要

关系数据库标准语言SQL-1

1 SQL概述

2 学生-课程数据库

3 数据定义

4 数据查询—单表

SQL-2

4 数据查询—多表

SQL-3

5 数据更新

6 空值的处理

7 视图

5. 数据更新

- 插入数据insert
  - INSERT INTO <表名> [(<属性列1>[,<属性列2 >…)]
    VALUES (<常量1> [,<常量2>]…);
- 修改数据update
  - UPDATE <表名> SET <列名>=<表达式>[,<列名>=<表达式>]… [WHERE <条件>];
- 删除数据delete
  - DELETE FROM <表名> [WHERE <条件>];

学生-课程 数据库

学生-课程模式 S-T：学生关系Student、课程关系Course和选修关系SC

Student

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

SC

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

基本表的定义

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS

[例3.5] 建立“学生”表Student。

学号是主码，姓名取值唯一。

```
CREATE TABLE Student
    (Sno   CHAR(9) PRIMARY KEY,
                          /* 列级完整性约束条件,Sno是主码*/
     Sname CHAR(20) UNIQUE,          /* Sname取唯一值*/
     Ssex   CHAR(2),
     Sage   SMALLINT,
     Sdept  CHAR(20)
    );
```

主码

UNIQUE 约束

表定义的注释

基本表的定义

[例3.6] 建立一个“课程”表Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4

```
CREATE TABLE  Course
      (Cno       CHAR(4) PRIMARY KEY,
       Cname   CHAR(40),
       Cpno     CHAR(4),
        Ccredit  SMALLINT，
        FOREIGN KEY (Cpno) REFERENCES  Course(Cno)
      );
```

先修课

Cpno是外码
被参照表是Course
被参照列是Cno

基本表的定义

[例3.7] 建立一个学生选课表SC

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

```
CREATE TABLE  SC
      (Sno  CHAR(9),
       Cno  CHAR(4),
       Grade  SMALLINT,
       PRIMARY KEY (Sno,Cno),
                /* 主码由两个属性构成，必须作为表级完整性进行定义*/
       FOREIGN KEY (Sno) REFERENCES Student(Sno),
                /* 表级完整性约束条件，Sno是外码，被参照表是Student */
       FOREIGN KEY (Cno)REFERENCES Course(Cno)
                /* 表级完整性约束条件， Cno是外码，被参照表是Course*/
);
```

学生-课程 S_T数据库

学生-课程模式 S-T：涵盖 一对一，一对多，多对多，自我参照 4种关系

Student

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

SC

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

插入数据

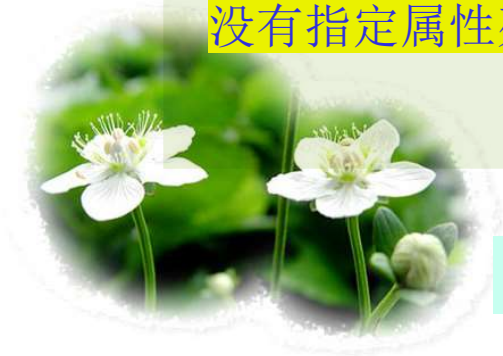
[例3.69]将一个新学生元组（学号：201215128;姓名：陈冬;性别：男;所在系：IS;年龄：18岁）插入到Student表中。

```
INSERT
INTO  Student (Sno,Sname,Ssex,Sdept,Sage)
VALUES ('201215128','陈冬','男','IS',18);
```

[例3.70]将学生张成民的信息插入到Student表中。

```
INSERT
INTO  Student
VALUES ('201215126','张成民','男',18,'CS');
```

没有指定属性列：表示要插入的是一条完整的元组，且属性列属性与表定义中的顺序一致



插入数据

[例3.71] 插入一条选课记录（ '200215128','1 '）。

```
INSERT
INTO SC(Sno,Cno)
VALUES ('201215128 ',' 1 ');
```

系统将在新插入记录的Grade列上自动地赋空值。

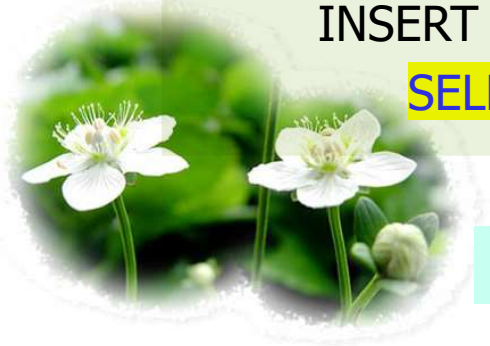
[例3.72] 对每一个系，求学生的平均年龄，并把结果存入数据库

第一步：建表

```
CREATE  TABLE  Dept_age ( Sdept     CHAR(15)      /*系名*/
    Avg_age SMALLINT);                  /*学生平均年龄*/
```

第二步：插入数据

```
INSERT   INTO  Dept_age(Sdept,Avg_age)
    SELECT  Sdept，AVG(Sage)  FROM Student  GROUP BY Sdept;
```



修改数据

[例3.73] 将学生201215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22   WHERE  Sno=' 201215121';
```

[例3.74] 将所有学生的年龄增加1岁。

```
UPDATE Student   SET Sage= Sage+1;
```

UPDATE -> SELECT

[例3.75] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC    SET     Grade=0
WHERE Sno  IN
      (SELECT Sno  FROM  Student  WHERE  Sdept= 'CS' );
```

删除数据

[例3.76] 删除学号为201215128的学生记录。

```
DELETE FROM Student
 WHERE Sno='201215128';
```

[例3.77] 删除所有的学生选课记录。

```
DELETE  FROM SC;
```

DELETE -> SELECT

[例3.78] 删除计算机科学系所有学生的选课记录。

```
DELETE  FROM  SC
WHERE  Sno  IN
    (SELECT  Sno FROM Student WHERE  Sdept= 'CS') ;
```

6. 空值的处理

- 空值就是“不知道”或“不存在”或“无意义”的值。
- 一般有以下几种情况：
  - 该属性应该有一个值，但目前不知道它的具体值
  - 该属性不应该有值
  - 由于某种原因不便于填写

- 空值是一个很特殊的值，含有不确定性。对关系运算带来特殊的问题，需要做特殊的处理。

空值的处理

[例 3.79]向SC表中插入一个元组，学生号是"201215126"，课程号是"1"，成绩为空。

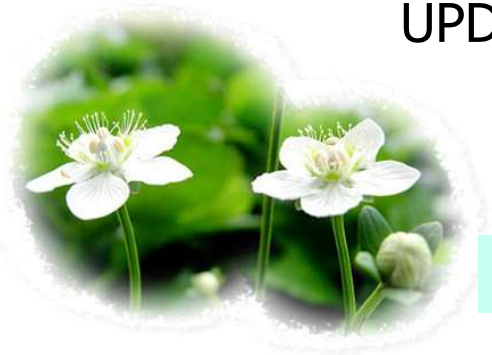
```
INSERT INTO SC(Sno,Cno,Grade)
VALUES('201215126 ','1',NULL);   /*该学生还没有考试成绩，取空值*/
```

或

```
INSERT INTO SC(Sno,Cno)
VALUES(' 201215126 ','1');          /*没有赋值的属性，其值为空值*/
```

[例3.80] 将Student表中学生号为"201215200"的学生所属的系改为空值。

```
UPDATE Student     SET Sdept = NULL
    WHERE Sno='201215200';
```



空值的处理

- 判断一个属性的值是否为空值，用IS NULL或IS NOT NULL来表示。

[例 3.81] 从Student表中找出漏填了数据的学生信息

```
SELECT *
FROM Student
WHERE Sname IS NULL OR Ssex IS NULL OR Sage IS NULL OR
      Sdept IS NULL;
```

空值的处理

- 空值与另一个值（包括另一个空值）的算术运算的结果为空值
- 空值与另一个值（包括另一个空值）的比较运算的结果为**UNKNOWN**。
  - 有**UNKNOWN**后，二值（**TRUE**，**FALSE**）逻辑就扩展成了三值逻辑

x	y	x AND y	x OR y	NOT x
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

空值的处理

[例3.82] 找出选修1号课程的不及格的学生。

```
SELECT Sno   FROM SC
WHERE Grade < 60 AND Cno='1';
```

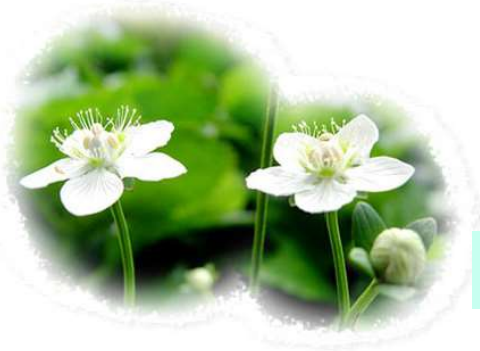
查询结果不包括缺考的学生，因为他们的Grade值为 null。

[例 3.83] 选出选修1号课程的不及格的学生以及缺考的学生。

```
SELECT Sno  FROM SC
WHERE Grade < 60 AND Cno='1'
UNION
SELECT Sno  FROM SC
WHERE Grade IS NULL AND Cno='1';
```

或者

```
SELECT Sno  FROM SC
WHERE Cno='1' AND (Grade<60 OR Grade IS NULL);
```



7. 视图VIEW

- 视图的特点
  - 虚表，是从一个或几个基本表（或视图）导出的表
  - 只存放视图的定义，不存放视图对应的数据
  - 基表中的数据发生变化，从视图中查询出的数据也随之改变

1 定义视图

2 查询视图

3 更新视图

4 视图的作用

视图

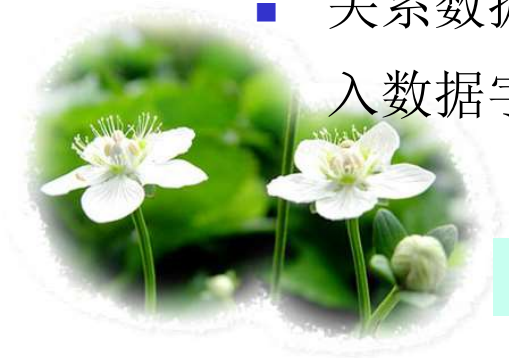
- CREATE VIEW

  <视图名> [(<列名> [,<列名>]...)]

  AS <子查询> [WITH CHECK OPTION];

  - WITH CHECK OPTION
    - 对视图进行UPDATE，INSERT和DELETE操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）
  - 子查询可以是任意的SELECT语句，是否可以含有ORDER BY子句和DISTINCT短语，则决定具体系统的实现。

- 关系数据库管理系统执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。



视图

[例3.84] 建立信息系学生的视图。

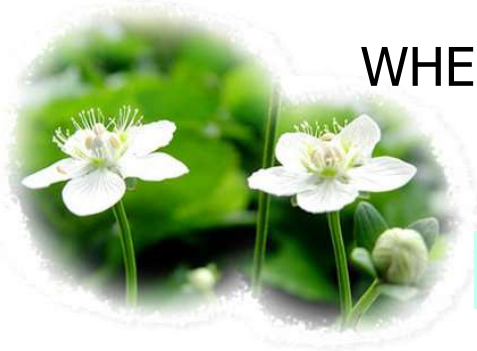
```
CREATE VIEW IS_Student  AS
    SELECT Sno,Sname,Sage
    FROM     Student
    WHERE  Sdept= 'IS';
```

- ✓ IS_Student是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了主码，我们称这类视图为行列子集视图。

[例3.85]建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该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;
```

- ✓ 定义IS_Student视图时加上了WITH CHECK OPTION子句，对该视图进行插入、修改和删除操作时，RDBMS会自动加上Sdept='IS'的条件。



视图

- 基于多个基表的视图

[例3.86] 建立信息系选修了1号课程的学生的视图（包括学号、姓名、成绩）。

```
CREATE VIEW IS_S1(Sno,Sname,Grade)
AS
SELECT Student.Sno,Sname,Grade
FROM  Student,SC
WHERE  Sdept= 'IS' AND
            Student.Sno=SC.Sno AND
            SC.Cno= '1';
```

视图

- 基于视图的视图

[例3.87] 建立信息系选修了1号课程且成绩在90分以上的学生的视图。

```
CREATE VIEW IS_S2
AS
SELECT Sno,Sname,Grade
FROM  IS_S1
WHERE  Grade>=90;
```

视图

- 带表达式的视图

[例3.88] 定义一个反映学生出生年份的视图。

```
CREATE  VIEW BT_S(Sno,Sname,Sbirth)
AS
SELECT Sno,Sname,2014-Sage  FROM  Student;
```

- 分组视图

[例3.89] 将学生的学号及平均成绩定义为一个视图

```
CREATE  VIEW S_G(Sno,Gavg)
 AS
 SELECT Sno,AVG(Grade)  FROM  SC
        GROUP BY Sno;
```

视图

[例3.90]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student(F_Sno,name,sex,age,dept)
AS
SELECT  *                          /*没有不指定属性列*/
FROM  Student
WHERE Ssex='女';
```

缺点：

修改基表Student的结构后，Student表与F_Student视图 的映象关系被破坏，导致该视图不能正确工作。

视图

- DROP VIEW <视图名>[CASCADE];
  - 该语句从数据字典中删除指定的视图定义
  - 如果该视图上还导出了其他视图，使用CASCADE级联删除语句，把该视图和由它导出的所有视图一起删除
  - 删除基表时，由该基表导出的所有视图定义都必须显式地使用DROP VIEW语句删除

```
CREATE VIEW IS_S2
    AS
    SELECT Sno,Sname,Grade
    FROM  IS_S1
    WHERE  Grade>=90;
```

[例3.91] 删除视图BT_S和IS_S1

```
DROP VIEW BT_S;          /*成功执行*/
DROP VIEW IS_S1;         /*拒绝执行*/
```

要删除IS_S1，需使用级联删除：

```
DROP VIEW IS_S1 CASCADE;
```

视图

[例3.92] 在信息系学生的视图中找出年龄小于20岁的学生。

```
SELECT  Sno,Sage     FROM     IS_Student
 WHERE  Sage<20;
```

- 视图消解转换后的查询语句为：

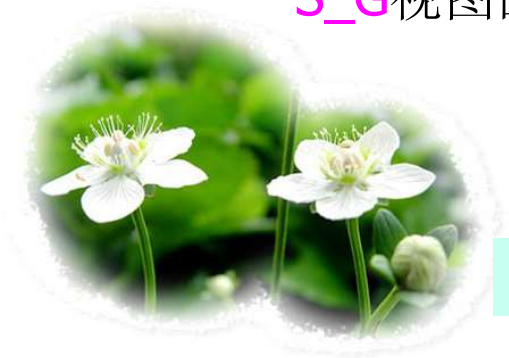
```
SELECT  Sno,Sage  FROM  Student
WHERE  Sdept= 'IS'  AND  Sage<20;
```

[例3.94]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  Gavg>=90;
```

S_G视图的子查询定义： CREATE VIEW S_G (Sno,Gavg) AS

```
SELECT  Sno,AVG(Grade) FROM  SC  GROUP BY Sno;
```



视图

[例3.94]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  Gavg>=90;
```

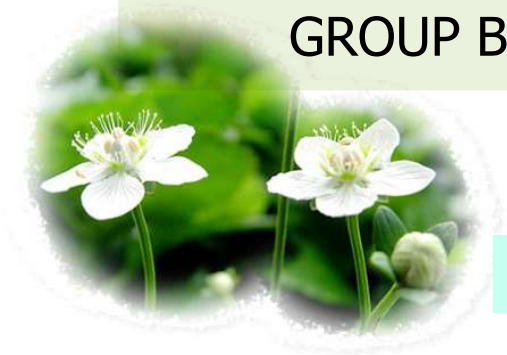
S_G视图的子查询定义： CREATE VIEW S_G (Sno,Gavg) AS
SELECT Sno,AVG(Grade) FROM SC GROUP BY Sno;

错误视图消解转换：

```
SELECT Sno，AVG(Grade)   FROM     SC
WHERE  AVG(Grade)>=90    GROUP BY Sno;
```

正确视图消解转换：

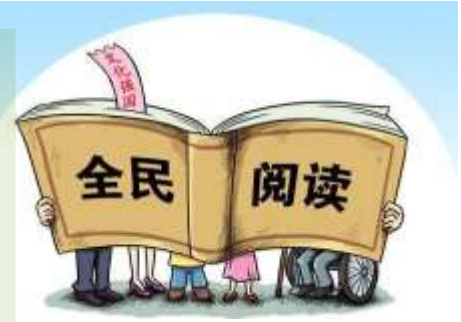
```
SELECT  Sno,AVG(Grade)  FROM  SC
GROUP BY Sno   HAVING AVG(Grade)>=90;
```



视图

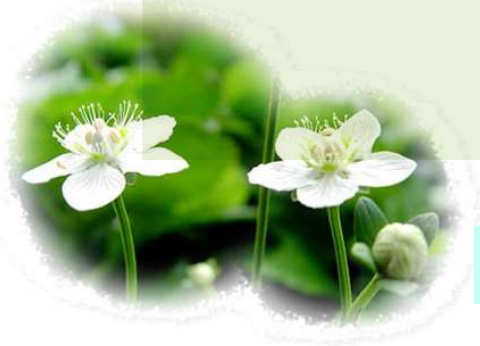
[例3.94]也可以用如下SQL语句完成

```
SELECT *
FROM  (SELECT Sno,AVG(Grade)
            FROM  SC
            GROUP BY Sno) AS S_G(Sno,Gavg)
WHERE Gavg>=90;
```



物化视图法：把视图查询结果作为一个表，然后再对表进行查询，达到查询视图的效果。

比较 视图消解法 与 物化视图法



视图 .更新

[例3.95] 将信息系学生视图IS_Student中学号”201215122”的学生姓名改为”刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= ' 201215122 ';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= ' 201215122 ' AND Sdept= 'IS';
```

视图 .更新

[例3.96] 向信息系学生视图IS_S中插入一个新的学生记录，其中学号为"201215129"，姓名为"赵新"，年龄为20岁

```
INSERT
INTO IS_Student
VALUES('201215129','赵新',20);
```

转换为对基本表的更新：

```
INSERT
INTO   Student(Sno,Sname,Sage,Sdept)
VALUES('200215129 ','赵新',20,'IS' );
```

视图 .更新

[例3.97]删除信息系学生视图IS_Student中学号为”201215129”的记录

```
DELETE
FROM IS_Student
WHERE Sno= ' 201215129 ';
```

转换为对基本表的更新：

```
DELETE
FROM Student
WHERE Sno= ' 201215129 ' AND Sdept= 'IS';
```

视图 . 更新

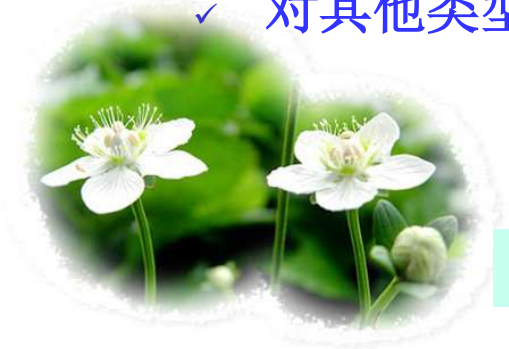
- 更新视图的限制：一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新

例：例3.89定义的视图S_G为不可更新视图。

```
UPDATE  S_G
SET     Gavg=90
WHERE   Sno= '201215121';
```

这个对视图的更新无法转换成对基本表SC的更新

- ✓ 允许对行列子集视图进行更新
- ✓ 对其他类型视图的更新不同系统有不同限制



视图 .作用

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

视图 .作用

- 视图能够简化用户的操作
  - 基于多张表连接形成的视图
  - 基于复杂嵌套查询的视图
  - 含导出属性的视图
- 视图能够对机密数据提供安全保护
  - 对不同用户定义不同视图，使每个用户只能看到他有权看到的数据
- 适当的利用视图可以更清晰的表达查询
  - 经常需要执行这样的查询“对每个同学找出他获得最高成绩的课程号”。可以先定义一个视图，求出每个同学获得的最高成绩

视图 .作用

- 视图对重构数据库提供了一定程度的逻辑独立性

例：学生关系Student(Sno,Sname,Ssex,Sage,Sdept)

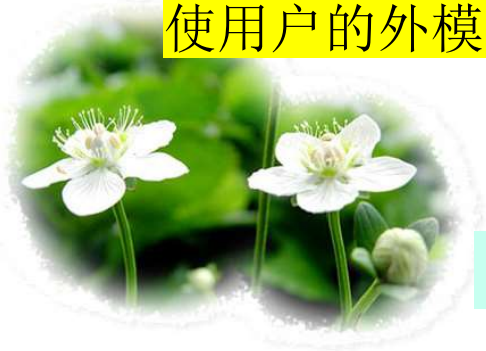
“垂直”地分成两个基本表：

SX(Sno,Sname,Sage)　　SY(Sno,Ssex,Sdept)

通过建立一个视图Student：

```
CREATE VIEW  Student(Sno,Sname,Ssex,Sage,Sdept)
AS
    SELECT  SX.Sno,SX.Sname,SY.Ssex,SX.Sage,SY.Sdept
    FROM  SX,SY    WHERE  SX.Sno=SY.Sno;
```

使用户的外模式保持不变，用户的应用程序通过视图仍然能够查找数据



小结

- **SQL可以分为**

  数据定义、数据查询

  数据更新、数据控制

- SQL是关系数据库语言的工业标准

  大部分数据库管理系统产品都能支持SQL92

- 数据对象：**模式，数据库，表，视图，索引**
  - 构造一个本地数据库
  - 合作尝试构造一个网络数据库