

## 3.4 数据查询

---

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

**3.4.4 集合查询**

3.4.5 基于派生表的查询

3.4.5 Select语句的一般形式

## 3.4.4 集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 参加集合操作的各查询结果的列数必须相同;对应项的数据类型也必须相同

# 集合查询（续）

[例 3.64] 查询计算机科学系的学生及（或）年龄不大于19岁的学生。

```
SELECT *  
FROM Student  
WHERE Sdept= 'CS'  
UNION  
SELECT *  
FROM Student  
WHERE Sage<=19;
```

- UNION: 将多个查询结果合并起来时，系统自动去掉重复元组
- UNION ALL: 将多个查询结果合并起来时，保留重复元组

# 集合查询（续）

[例 3.65] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno
```

```
FROM SC
```

```
WHERE Cno=' 1 '
```

```
UNION
```

```
SELECT Sno
```

```
FROM SC
```

```
WHERE Cno= ' 2 ';
```

## 集合查询（续）

[例3.66] 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
```

## 集合查询（续）

[例 3.66] 实际上就是查询计算机科学系中年龄不大于19岁的学生。

```
SELECT *
```

```
FROM Student
```

```
WHERE Sdept= 'CS' AND  Sage<=19;
```

## 集合查询（续）

[例 3.67]查询既选修了课程1又选修了课程2的学生。

```
SELECT Sno  
FROM SC  
WHERE Cno='1 '  
INTERSECT  
SELECT Sno  
FROM SC  
WHERE Cno='2 ';
```

## 集合查询（续）

[例3.67] 查询既选修了课程1又选修了课程2的学生。

也可以表示为：

```
SELECT Sno
FROM SC
WHERE Cno=' 1 ' AND Sno IN
        (SELECT Sno
         FROM SC
         WHERE Cno=' 2 ');
```



## 集合查询（续）

[例 3.68] 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;
```

## 集合查询（续）

[例3.68]实际上是查询计算机科学系中年龄大于19岁的学生

```
SELECT *
```

```
FROM Student
```

```
WHERE Sdept= 'CS' AND  Sage>19;
```

## 3.4 数据查询

---

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

3.4.4 集合查询

**3.4.5 基于派生表的查询**

3.4.5 Select语句的一般形式

### 3.4.5 基于派生表的查询

- 子查询不仅可以出现在WHERE子句中，还可以出现在FROM子句中，这时子查询生成的临时派生表（Derived Table）成为主查询的查询对象。

[例3.57]找出每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade)
          FROM SC
          GROUP BY Sno)
      AS Avg_sc(avg_sno,avg_grade)
WHERE SC.Sno = Avg_sc.avg_sno
      and SC.Grade >=Avg_sc.avg_grade
```

## 基于派生表的查询（续）

- 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询SELECT子句后面的列名为其缺省属性。

[例3.60]查询所有选修了1号课程的学生姓名，可以用如下查询完成：

```
SELECT Sname
FROM   Student,
       (SELECT Sno FROM SC WHERE Cno=' 1 ') AS SC1
WHERE  Student.Sno=SC1.Sno;
```

## 3.4 数据查询

---

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

3.4.4 集合查询

3.4.5 基于派生表的查询

**3.4.6 Select语句的一般形式**

## 3. 4. 6 SELECT语句的一般格式

**SELECT** [ALL|DISTINCT]

<目标列表达式> [别名] [ ,<目标列表达式> [别名]] ...

**FROM** <表名或视图名> [别名]

[ ,<表名或视图名> [别名]] ...

|(<SELECT语句>)[AS]<别名>

[**WHERE** <条件表达式>]

[**GROUP BY** <列名1>[**HAVING**<条件表达式>]]

[**ORDER BY** <列名2> [ASC|DESC]];

# 1. 目标列表表达式的可选格式

## ● 目标列表表达式格式

(1) \*

(2) <表名>.\*

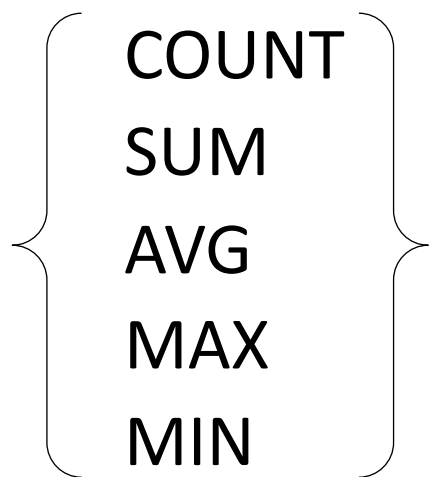
(3) COUNT([DISTINCT|ALL]\* )

(4) [<表名>.]<属性列名表达式>[,<表名>.]<属性列名表达式>]...

其中<属性列名表达式>可以是由属性列、作用于属性列的聚集函数和常量的任意算术运算 (+, -, \*, /) 组成的运算公式



## 2. 聚集函数的一般格式

COUNT  
SUM  
AVG  
MAX  
MIN

([DISTINCT|ALL] <列名>)

### 3. WHERE子句的条件表达式的可选格式

(1)

$\langle \text{属性列名} \rangle \theta \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ [\text{ANY}|\text{ALL}] \text{ (SELECT语句)} \end{array} \right\}$

(2)

$\langle \text{属性列名} \rangle [\text{NOT}] \text{BETWEEN} \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ (\text{SELECT语句}) \end{array} \right\} \text{AND} \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ (\text{SELECT语句}) \end{array} \right\}$

## WHERE子句的条件表达式格式（续）

(3)  $\langle \text{属性列名} \rangle [\text{NOT}] \text{IN} \left\{ \begin{array}{l} (\langle \text{值1} \rangle [, \langle \text{值2} \rangle ] \dots) \\ (\text{SELECT语句}) \end{array} \right\}$

(4)  $\langle \text{属性列名} \rangle [\text{NOT}] \text{LIKE} \langle \text{匹配串} \rangle$

(5)  $\langle \text{属性列名} \rangle \text{IS} [\text{NOT}] \text{NULL}$

(6)  $[\text{NOT}] \text{EXISTS} (\text{SELECT语句})$

# WHERE子句的条件表达式格式（续）

(7)

$$\langle \text{条件表达式} \rangle \left\{ \begin{array}{c} \text{AND} \\ \text{OR} \end{array} \right\} \langle \text{条件表达式} \rangle \left\{ \begin{array}{c} \text{AND} \\ \text{OR} \end{array} \right\} \langle \text{条件表达} \rangle \dots$$

# 第三章 关系数据库标准语言SQL

---

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

**3.5 数据更新**

3.6 空值的处理

3.7 视图

3.8 小结

## 3.5 数据更新

---

### 3.5.1 插入数据

### 3.5.2 修改数据

### 3.5.3 删除数据

## 3.5.1 插入数据

- 两种插入数据方式
  - 插入元组
  - 插入子查询结果
    - ✓ 可以一次插入多个元组

# 1. 插入元组

- 语句格式

INSERT

INTO <表名> [(<属性列1>[,<属性列2 >...])]

VALUES (<常量1> [,<常量2>]... );

- 功能

- 将新元组插入指定表中



# 插入元组（续）

## ● INTO子句

➤ 指定要插入数据的表名及属性列

✓ 属性列的顺序可与表定义中的顺序不一致

➤ 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

➤ 指定部分属性列：插入的元组在其余属性列上取空值

# 插入元组（续）

- VALUES子句

- 提供的值必须与INTO子句匹配

- ✓值的个数

- ✓值的类型

## 插入元组（续）

[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);
```

# 插入元组（续）

[例3.71] 插入一条选课记录（ '200215128','1 '）。

```
INSERT
```

```
INTO SC(Sno,Cno)
```

```
VALUES ('201215128 ','1 ');
```

关系数据库管理系统将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT
```

```
INTO SC
```

```
VALUES (' 201215128 ','1 ',NULL);
```

# 插入元组（续）

[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');
```

## 2. 插入子查询结果

- 语句格式

INSERT

INTO <表名> [(<属性列1> [,<属性列2>... ]]

子查询;

- INTO子句

- 子查询

- SELECT子句目标列必须与INTO子句匹配

- 值的个数

- 值的类型

## 插入子查询结果（续）

[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;
```

## 插入子查询结果（续）

- 关系数据库管理系统在执行插入语句时会检查所插元组是否破坏表上已定义的完整性规则
  - 实体完整性
  - 参照完整性
  - 用户定义的完整性
    - NOT NULL约束
    - UNIQUE约束
    - 值域约束



## 3.5 数据更新

---

### 3.5.1 插入数据

### 3.5.2 修改数据

### 3.5.3 删除数据

## 3.5.2 修改数据

### ● 语句格式

UPDATE <表名>

SET <列名>=<表达式>[,<列名>=<表达式>]...

[WHERE <条件>];

### ● 功能

- 修改指定表中满足WHERE子句条件的元组
- SET子句给出<表达式>的值用于取代相应的属性列
- 如果省略WHERE子句，表示要修改表中的所有元组

# 修改数据（续）

- 三种修改方式

- 修改某一个元组的值
- 修改多个元组的值
- 带子查询的修改语句

# 1. 修改某一个元组的值

[例3.73] 将学生201215121的年龄改为22岁

```
UPDATE Student
```

```
SET Sage=22
```

```
WHERE Sno=' 201215121 ';
```

## 2. 修改多个元组的值

[例3.74] 将所有学生的年龄增加1岁。

```
UPDATE Student
```

```
SET Sage= Sage+1;
```

/\*省略WHERE，意味着所有元组都要修改\*/

### 3. 带子查询的修改语句

[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  
      (SELETE Sno  
       FROM   Student  
       WHERE  Sdept= 'CS' );
```

# 修改数据（续）

- 关系数据库管理系统在执行修改语句时会检查修改操作是否破坏表上已定义的完整性规则
  - 实体完整性
  - 参照完整性
  - 用户定义的完整性
    - NOT NULL约束
    - UNIQUE约束
    - 值域约束

## 3.5 数据更新

---

### 3.5.1 插入数据

### 3.5.2 修改数据

### 3.5.3 删除数据



## 3.5.3 删除数据

- 语句格式

DELETE

FROM <表名>

[WHERE <条件>];

- 功能

- 删除指定表中满足WHERE子句条件的元组

- WHERE子句

- 指定要删除的元组

- 缺省表示要删除表中的所有元组，表的定义仍在字典中

# 删除数据（续）

- 三种删除方式

- 删除某一个元组的值

- 删除多个元组的值

- 带子查询的删除语句

# 1. 删除某一个元组的值

[例3.76] 删除学号为201215128的学生记录。

```
DELETE
```

```
FROM Student
```

```
WHERE Sno= '201215128';
```

## 2. 删除多个元组的值

[例3.77] 删除所有的学生选课记录。

```
DELETE
```

```
FROM SC;
```

### 3. 带子查询的删除语句

[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
```

```
    (SELETE Sno
```

```
    FROM Student
```

```
    WHERE Sdept= 'CS') ;
```

# 第三章 关系数据库标准语言SQL

---

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

**3.6 空值的处理**

3.7 视图

3.8 小结

## 3.6 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

# 1. 空值的产生

- 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。

- 空值的产生

[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*/
```



## 空值的产生（续）

[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';
```

## 2. 空值的判断

- 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;
```

### 3. 空值的约束条件

- 属性定义（或者域定义）中
  - 有NOT NULL约束条件的不能取空值
  - 加了UNIQUE限制的属性不能取空值
  - 码属性不能取空值

## 4. 空值的算术运算、比较运算和逻辑运算

- 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，传统二值（TRUE，FALSE）逻辑就扩展成了三值逻辑

# 空值的算术运算、比较运算和逻辑运算 (续)

表3.8 逻辑运算符真值表

<b>x</b>	<b>y</b>	<b>x AND y</b>	<b>x OR y</b>	<b>NOT x</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>U</b>	<b>U</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>U</b>	<b>T</b>	<b>U</b>	<b>T</b>	<b>U</b>
<b>U</b>	<b>U</b>	<b>U</b>	<b>U</b>	<b>U</b>
<b>U</b>	<b>F</b>	<b>F</b>	<b>U</b>	<b>U</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>U</b>	<b>F</b>	<b>U</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>T</b>

T表示TRUE， F表示FALSE， U表示UNKNOWN

# 空值的算术运算、比较运算和逻辑运算 (续)

[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。

# 空值的算术运算、比较运算和逻辑运算 (续)

[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);
```

# 第三章 关系数据库标准语言SQL

---

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

3.6 空值的处理

**3.7 视图**

3.8 小结



## 3.7 视图

### ●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

## 3.7 视图

---

### 3.7.1 定义视图

### 3.7.2 查询视图

### 3.7.3 更新视图

### 3.7.4 视图的作用

## 3.7.1 定义视图

1.建立视图

2.删除视图

# 1. 建立视图

- 语句格式

CREATE VIEW

<视图名> [(<列名> [,<列名>]...)]

AS <子查询>

[WITH CHECK OPTION];

# 建立视图（续）

- 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。

# 建立视图（续）

- 组成视图的属性列名：全部省略或全部指定

- 全部省略：

- ✓由子查询中SELECT目标列中的诸字段组成

- 明确指定视图的所有列名：

- ✓某个目标列是聚集函数或列表表达式

- ✓多表连接时选出了几个同名列作为视图的字段

- ✓需要在视图中为某个列启用新的更合适的名字

## 建立视图（续）

- 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。
- 在对视图查询时，按视图的定义从基本表中将数据查出。

## 建立视图（续）

[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';
```





## 建立视图（续）

[例3.85]建立信息系学生的视图，并要求对该视图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;
```

## 建立视图（续）

- 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**，**关系数据库管理系统会自动加上Sdept='IS'的条件**。
- 若一个视图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**。
  - IS\_Student视图就是一个行列子集视图。

## 建立视图（续）

- 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';
```

# 建立视图（续）

## ●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;
```

# 建立视图（续）

- 带表达式的视图

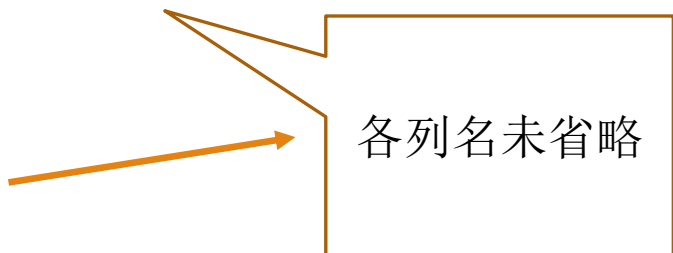
[例3.88] 定义一个反映学生出生年份的视图。

```
CREATE VIEW BT_S(Sno, Sname, Sbirth)
```

```
AS
```

```
SELECT Sno,Sname,2014-Sage
```

```
FROM Student;
```



各列名未省略

# 建立视图（续）

## ● 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

```
CREAT VIEW S_G(Sno,Gavg)
```

```
AS
```

```
SELECT Sno, AVG(Grade)
```

```
FROM SC
```

```
GROUP BY Sno;
```

各列名未省略



# 建立视图（续）

[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。

## 2. 删除视图

- 语句的格式:

**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。

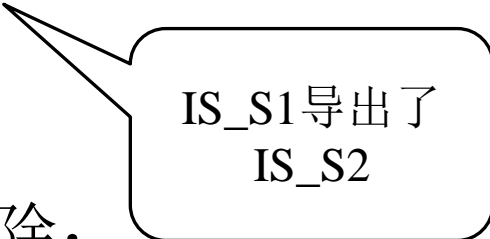


## 删除视图（续）

[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\*/



IS\_S1导出了  
IS\_S2

要删除IS\_S1，需使用级联删除：

DROP VIEW IS\_S1 **CASCADE**;

/\*DROP VIEW中在SQL SERVER中没有CASCADE选项\*/

## 3.7 视图

---

### 3.7.1 定义视图

### 3.7.2 查询视图

### 3.7.3 更新视图

### 3.7.4 视图的作用

## 3.7.2 查询视图

- 用户角度：查询视图与查询基本表相同
- 关系数据库管理系统实现视图查询的方法
  - 视图消解法（View Resolution）
    - ✓ 进行有效性检查
    - ✓ 转换成等价的对基本表的查询
    - ✓ 执行修正后的查询

## 查询视图（续）

[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;
```



## 查询视图（续）

[例3.93] 查询选修了1号课程的信息系学生

```
SELECT IS_Student.Sno,Sname  
FROM   IS_Student,SC  
WHERE  IS_Student.Sno =SC.Sno AND SC.Cno= '1';
```

# 查询视图（续）

- 视图消解法的局限

- 有些情况下，视图消解法不能生成正确的查询。

[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;
```

## 查询视图（续）

错误：

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

正确：

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

## 查询视图（续）

[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;
```



## 3.7 视图

---

### 3.7.1 定义视图

### 3.7.2 查询视图

### 3.7.3 更新视图

### 3.7.4 视图的作用

## 更新视图（续）

[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';
```

## 更新视图（续）

[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    /* P122 例3.85为 IS_Student的定义*/  
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' );
```

## 更新视图（续）

[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';
```

## 更新视图（续）

- 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，因为系统无法修改各科成绩，以使平均成绩为90。

## 更新视图（续）

- 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

## 更新视图（续）

- DB2（IBM公司的DBMS）对视图更新的限制：
  - ① 若视图是由两个以上基本表导出的，则此视图不允许更新。
  - ② 若视图的字段来自字段表达式或常数，则不允许对此视图执行INSERT和UPDATE操作，但允许执行DELETE操作。
  - ③ 若视图的字段来自聚集函数，则此视图不允许更新。
  - ④ 若视图定义中含有GROUP BY子句，则此视图不允许更新。
  - ⑤ 若视图定义中含有DISTINCT短语，则此视图不允许更新。
  - ⑥ 若视图定义中有嵌套查询，并且内层查询的FROM子句中涉及的表也是导出该视图的基本表，则此视图不允许更新。

## 更新视图（续）

例：将SC中成绩在平均成绩之上的元组定义成一个视图

```
CREATE VIEW GOOD_SC  
AS  
SELECT Sno,Cno,Grade  
FROM SC  
WHERE Grade >  
      (SELECT AVG(Grade)  
       FROM SC);
```

⑦一个不允许更新的视图上定义的视图也不允许更新



## 3.7 视图

---

### 3.7.1 定义视图

### 3.7.2 查询视图

### 3.7.3 更新视图

### 3.7.4 视图的作用

## 3.7.4 视图的作用

- 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

# 视图的作用（续）

- 视图能够简化用户的操作

当视图中数据不是直接来自基本表时，定义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

# 视图的作用（续）

- 视图使用户能以多种角度看待同一数据
  - 视图机制能使不同用户以不同方式看待同一数据，适应数据库共享的需要

# 视图的作用（续）

- 视图对重构数据库提供了一定程度的逻辑独立性

➤ 数据库重构：

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)

## 视图的作用（续）

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;
```

SX、SY两个表，应用程序不用改，因为新建立的视图定义为用户原来的关系（SX与SY自然连接为原表STUDENT），使用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。

# 视图的作用（续）

- 视图对重构数据库提供了一定程度的逻辑独立性(续)
  - 视图只能在一定程度上提供数据的逻辑独立性
    - ✓ 由于对视图的更新是有条件的，因此应用程序中修改数据的语句可能仍会因基本表结构的改变而改变。
- 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

# 视图的作用（续）

- 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

```
CREATE VIEW VMGRADE
```

```
AS
```

```
SELECT Sno, MAX(Grade) Mgrade
```

```
FROM SC
```

```
GROUP BY Sno;
```



## 视图的作用（续）

然后用如下的查询语句完成查询：

```
SELECT SC.Sno,Cno  
FROM SC,VMGRADE  
WHERE SC.Sno=VMGRADE.Sno AND  
SC.Grade=VMGRADE .Mgrade;
```

# 第三章 关系数据库标准语言SQL

---

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

3.6 空值的处理

3.7 视图

3.8 小结

## 3.8 小结

- SQL可以分为数据定义、数据查询、数据更新、数据控制四大部分
- 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,但是许多数据库系统只支持SQL99、SQL2008、SQL2011、SQL 2016的部分特征，至今尚没有一个数据库系统能够完全支持SQL99以上的标准。