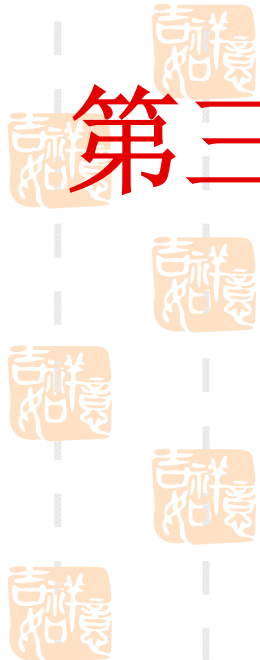




数据库系统概论

An Introduction to Database System

第三章 关系数据库标准语言SQL (续2)



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 视图

3.7 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

1. 插入元组

2. 插入子查询结果

➤ 可以一次插入多个元组



一、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）

[例1] 将一个新学生元组（学号：200215128；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 ('200215128', '陈冬', '男', 'IS', 18);

插入元组（续）

[例2] 将学生张成民的信息插入到Student表中。

```
INSERT
```

```
    INTO Student
```

```
VALUES ('200215126', '张成民', '男',  
18, 'CS');
```

插入元组（续）

[例3] 插入一条选课记录('200215128', '1')。

```
INSERT INTO SC(Sno, Cno)
VALUES ( '200215128 ', ' 1 ' );
```

RDBMS将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT INTO SC
VALUES ( '200215128 ', ' 1 ', NULL );
```


二、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;



■ 功能

将子查询结果插入指定表中



插入子查询结果（续）



- INTO子句(与插入元组类似)
- 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4] 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*/
```

插入子查询结果（续）



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）



RDBMS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



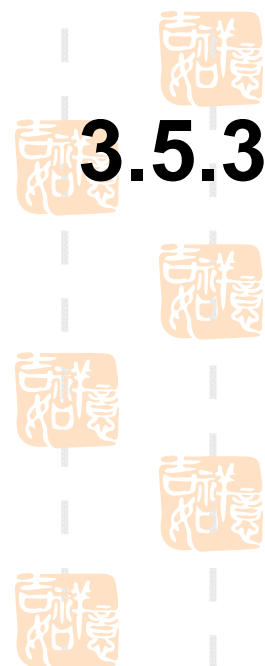
3.5 数据更新



3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据



3.4.2 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



修改数据（续）



■ 三种修改方式

1. 修改某一个元组的值

2. 修改多个元组的值

3. 带子查询的修改语句



1. 修改某一个元组的值

[例5] 将学生200215121的年龄改为22岁

```
UPDATE Student
```

```
SET Sage=22
```

```
WHERE Sno=' 200215121 ';
```

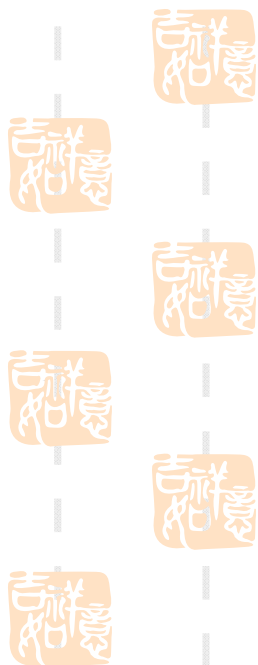
2. 修改多个元组的值



[例6] 将所有学生的年龄增加1岁

UPDATE Student

SET Sage= Sage+1;



3. 带子查询的修改语句

[例7] 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);
```

或:

```
UPDATE SC SET Grade=0
WHERE sno in
      (SELECT sno
       FROM Student
       WHERE sdept= 'CS');
```

修改数据（续）

RDBMS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

- 主码不允许修改

- 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全部元组，表的定义仍在字典中



删除数据（续）



■ 三种删除方式

1. 删除某一个元组的值

2. 删除多个元组的值

3. 带子查询的删除语句



1. 删除某一个元组的值

[例8] 删除学号为200215128的学生记录。

DELETE

FROM Student

WHERE Sno= '200215128 ';

2. 删除多个元组的值



[例9] 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10] 删除计算机科学系所有学生的选课记录。

```
DELETE FROM SC
```

```
WHERE 'CS' =
```

```
(SELECT Sdept
```

```
FROM Student
```

```
WHERE Student.Sno=SC.Sno);
```

或:

```
DELETE FROM SC
```

```
WHERE sno in
```

```
(SELECT 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 视图

3.7 小结

3.6 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
- (一般情况下)视图可以象基本表那样使用
- DBMS将视图转换成对基本表的操作



3.6 视图



基于视图的操作

- 查询

- 删除

- 受限更新

- 定义基于该视图的新视图



3.6 视图



3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用



3.6.1 定义视图



- 建立视图

- 删除视图



一、建立视图

- 语句格式

CREATE VIEW

<视图名> [(<列名> [, <列名>]...)]

AS <子查询>

[WITH CHECK OPTION];

- 组成视图的属性列名：全部省略或全部指定

- 子查询不允许含有ORDER BY子句和DISTINCT短语

建立视图（续）



- RDBMS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1] 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2]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;
```



建立视图（续）



对IS_Student视图的更新操作：

- 修改操作：自动加上Sdept= 'IS'的条件
- 删除操作：自动加上Sdept= 'IS'的条件
- 插入操作：自动检查Sdept属性值是否为'IS'

➤ 如果不是，则拒绝该插入操作

➤ 如果没有提供Sdept属性值，则自动定义Sdept为'IS'

建立视图（续）



- 基于多个基表的视图

[例3] 建立信息系选修了1号课程的学生视图。

```
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4] 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5] 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, 2010-Sage
FROM Student;
```



建立视图（续）



■ 分组视图

[例6] 将学生的学号及他的平均成绩定义为一个视图

假设SC表中“成绩”列Grade为数字型

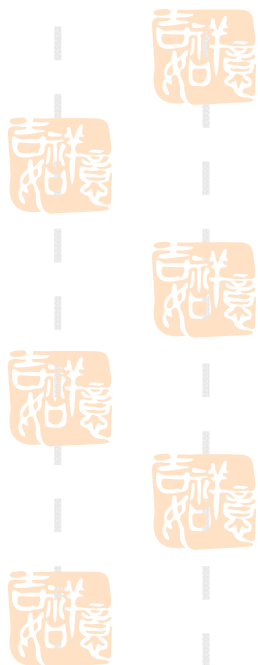
```
CREAT VIEW S_G(Sno, Gavg)
```

```
AS
```

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

```
FROM SC
```

```
GROUP BY Sno;
```



建立视图（续）

- 不指定属性列

[例7]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 *  
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。

二、删除视图

- 语句的格式:

DROP VIEW <视图名>;

- 该语句从数据字典中删除指定的视图定义, 不影响基本表

- 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

删除视图(续)



[例8] 删除视图BT_S:

```
DROP VIEW BT_S;
```

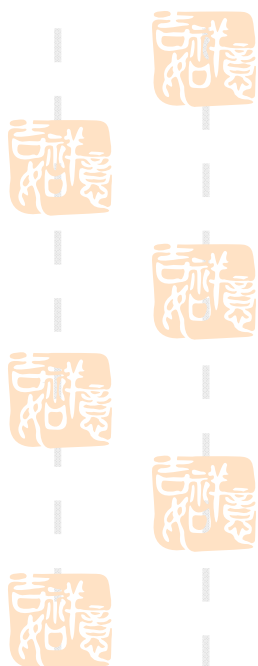
删除视图IS_S1:

```
DROP VIEW IS_S1;
```

➤ 拒绝执行

➤ 级联删除:

```
DROP VIEW IS_S1 CASCADE;
```



3.6 视图



3.6.1 定义视图

3.6.2 查询视图



3.6.3 更新视图



3.6.4 视图的作用



3.6.2 查询视图



- 用户角度：查询视图与查询基本表相同
- RDBMS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



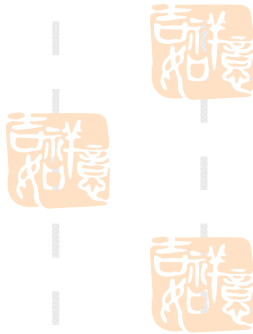
- 执行修正后的查询



查询视图（续）

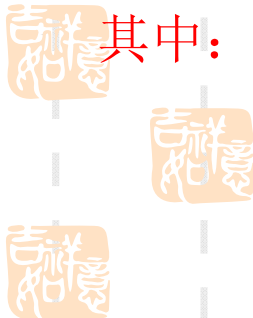


[例9] 在信息系学生的视图中找出年龄小于20岁的学生。



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

其中: CREATE VIEW IS_Student
AS



```
SELECT Sno, Sname, Sage FROM Student  
WHERE Sdept= 'IS';
```

查询视图（续）

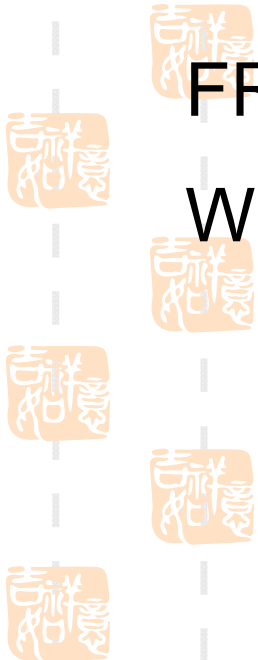


视图消解转换后的查询语句为：

SELECT Sno, Sage

FROM Student

WHERE Sdept= 'IS' AND Sage<20;



查询视图（续）

[例10] 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）

[例11]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;
```

问题： 如何用基本表做这个查询？

查询转换



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;
```



3.6 视图



3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用



更新视图（续）

- 一般而言，单表、原始属性构成的视图可以更新（可转换成对基本表的操作）

[例12] 将信息系学生视图IS_Student中学号200215122的学生姓名改为“刘辰”。

```
UPDATE IS_Student  
SET Sname= '刘辰'  
WHERE Sno= '200215122';
```

转换后的语句：

```
UPDATE Student  
SET Sname= '刘辰'  
WHERE Sno= '200215122 ' AND Sdept= 'IS';
```

更新视图（续）

[例13] 向信息系学生视图IS_S中插入一个新的学生记录：200215129，赵新，20岁

```
INSERT  
INTO IS_Student  
VALUES('95029', '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14]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 ';

转换为对基本表的更新:

DELETE

FROM Student

WHERE Sno= ' 200215129 ' 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：视图S_G为不可更新视图。

```
UPDATE S_G  
SET      Gavg=90  
WHERE Sno= '200215121';
```

这个对视图的更新无法转换成对基本表SC的更新

更新视图的限制

- 视图是不实际存储数据的虚表，对视图的更新最终要转换为对基本表的更新。
- 对视图的更新不能唯一地有意义地转换成为对相应基本表的更新时，不能更新视图。
- 一般的限制：
 - 由两个以上的基本表导出
 - 视图的字段来自字段表达式或常数
 - 包含 **GROUP BY, DISTINCT** 或聚集函数
 - 包含嵌套查询

3.6 视图



3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用



查询选修课程最多的学生学号

```
create view sno_num(sno,scount)
as
select sno,count(*) from sc group by sno
```

```
select sno from sno_num
where scount=(select max(scount) from
sno_num)
```

3.6.4 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



复杂查询实现方法1——WITH子句

- 在复杂查询中，将查询中相对独立的部分作为查询的中间结果，定义为临时视图。
- 临时视图不记入数据目录，仅用于附在临时视图定义后的查询语句中。查询语句结束后，临时视图也随之消失。

sc2定义了计算机系每门课程的最低和最高成绩。

- WITH sc2(cno,mingrade,maxgrade) AS
(SELECT cno, Min(grade), Max(grade)
FROM SC
WHERE cno like 'CS%' and grade is not null
GROUP BY cno)
- SELECT avg(mingrade),avg(maxgrade)
FROM sc2;

复杂查询实现方法2——AS子句

■SQL允许FROM子句使用子查询表达式，表达式产生的派生关系用AS子句完成。

下面查询表示 查询计算机类课程最低成绩和最高成绩的平均值。

■SELECT AVG(mingrade), avg(maxgrade)

■FROM (

SELECT cno, Min(grade), Max(grade)

FROM SC

WHERE cno like 'CS%' and grade is not null

GROUP BY cno) AS sc2 (cno,mingrade,maxgrade)

综合练习

- (1) 查询至少选修一门电机系课程的女生的姓名(假设电机系课程以ee号开头);
- (2) 查询每位学生已选修课程的门数和总平均成绩;
- (3) 查询每门课程选课的学生人数,最高成绩,最低成绩和平均成绩;
- (4) 查询所有课程的成绩都在80分以上的学生的姓名、学号、且按学号升序排列;
- (5) 查询缺成绩的学生的姓名, 缺成绩的课程号及其学分数;
- (6) 查询有一门以上(含一门)三个学分以上课程的成绩低于70分的学生
的姓名;
- (7) 查询18-20岁的学生的姓名,总平均成绩及已修学分数。
- (8) 在SC关系中, 删去SNO以' 01'开头的
所有记录。
- (9) 在student关系中增加以下记录:
 <0409101 何平 女 计算机系 18 >
- (10) 将课程号CS-221的学分数改为 3

综合练习

(1) 查询至少选修一门电机系课程的女生的姓名(假设电机系课程号以ee开头); ;

```
select SNAME from STUDENT where SSEX='女' and  
SNO in (select SNO from SC where CNO like 'ee%');
```

或:

```
select SNAME from STUDENT where SSEX='女'  
and exists
```

```
(select * from SC where SNO=STUDENT.SNO and  
CNO like 'ee%');
```

或:

```
select distinct SNAME from STUDENT,SC where  
STUDENT.SNO= SC.SNO and SSEX='女' and CNO  
like 'ee%';
```



(2) 查询每位学生已选修课程的门数和总平均成绩;

```
select SNO,count(*),avg(GRADE)
from SC
group by SNO;
```

(3) 查询每门课程选课的学生人数,最高成绩,最低成绩和平均成绩;

```
select CNO, count(*) 学生人数, max(GRADE) 最高成绩,
min(GRADE) 最低成绩, avg(GRADE) 平均成绩
from SC
group by CNO;
```


(4) 查询所有课程的成绩都在80分以上的学生的姓名、学号、且按学号升序排列;

```
select SNAME,SNO from STUDENT
where SNO not in (select SNO from SC
where GRADE <80 or GRADE is null) order by SNO;
```

或:

```
select SNAME,SNO from STUDENT where not exists
(select * from SC where SNO= STUDENT. SNO and
(GRADE<80 or GRADE is null )) order by SNO;
```

或: 如果不考虑成绩为空的情况

```
select SNAME,SNO from STUDENT where SNO in
(select sno from SC group by sno having
min(grade)>80 ) order by SNO;
```



(5) 查询缺成绩的学生的姓名，缺成绩的课程号及其学分数；

```
select SNAME,SC.CNO,CCREDIT  
from STUDENT,COURSE,SC  
where STUDENT.SNO=SC.SNO  
and COURSE.CNO=SC.CNO  
and GRADE is null;
```



(6) 查询有一门以上(含一门)三个学分以上课程的成绩低于70分的学生的姓名;

```
select SNAME from STUDENT where SNO in  
  (select SNO from SC where GRADE<70  
    and CNO in (select CNO from COURSE  
                where CCREDIT>=3));
```

或:

```
select distinct SNAME from STUDENT,COURSE,SC  
where STUDENT.SNO=SC.SNO and  
COURSE.CNO=SC.CNO and GRADE<70  
and CCREDIT>=3;
```



(7) 查询18-20岁的学生的姓名,总平均成绩及已修学分数。

```
Select STUDENT.SNO, SNAME, avg(GRADE) , sum_ccredit from
STUDENT,SC,COURSE,
(select SNO, sum(CCREDIT) from SC,COURSE
where COURSE.CNO=SC.CNO and grade>=60 group by SNO ) as
SSUM(sno,sum_ccredit)
where STUDENT.SNO=SC.SNO and COURSE.CNO=SC.CNO
and sage between 18 and 20 and sc.sno=SSUM.sno group by
STUDENT.SNO, SNAME;
```



(8) 在SC关系中，删去SNO以' 01'开头的所有记录。

```
delete from SC where SNO like '01%';
```

(9) 在student关系中增加以下记录：

<0409101 何平 女 计算机系 18 >

```
insert into STUDENT
```

```
values('0409101','何平','女','计算机系',18);
```

(10) 将课程CS-221的学分数改为 3

```
update COURSE set CCREDIT=3
```

```
where CNO='CS-221';
```

作业



- 3, 4, 5 (6) - (10), 7, 8, 9

