UNIT 8

SQL更新语句、视图与完整性

本讲主要目标

学完本讲后，你应该能够了解：

1、三个SQL更新语句的使用；

2、视图是虚表，视图在定义时，并没有真正执行定义语句中的查询语句；用户可以像使用基本表一样对视图进行查询和更新操作，但有时候是有限制的；

3、视图的作用；

4、SQL的更新操作可能破坏数据库的一致性；SQL提供了完整性约束的定义功能。

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

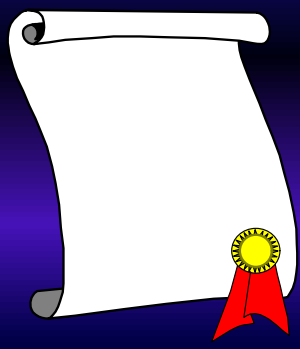
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



DreamHome 租赁数据库

DreamHome 案例的部分关系模式：

- **Branch** (branchNo, street, city, postcode)
- **Staff** (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
- **PropertyForRent** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
- **Client** (clientNo, fName, lName, telNo, prefType, maxRent)
- **PrivateOwner** (ownerNo, fName, lName, address, telNo)
- **Viewing** (clientNo, propertyNo, viewDate, comment)
- **Registration** (clientNo, branchNo, staffNo, dateJoined)

DreamHome租赁数据库实例:

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

PrivateOwner

ownerNo	fName	lName	address	telNo	eMail	password
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	jkeogh@lhh.com	********
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	cfarrel@gmail.com	********
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	tinam@hotmail.com	********
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	tony.shaw@ark.com	********

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-13
CR56	B003	SG37	11-Apr-12
CR74	B003	SG37	16-Nov-11
CR62	B007	SA9	7-Mar-12

一个学生-课程数据库

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

SC

学号 S#	课程号 C#	成绩 G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

C

课程号 C#	课程名 CN	先行课 CP#	学分 CC
C1	数据库	C5	4
C2	数学		2
C3	信息系统	C1	4
C4	操作系统	C6	3
C5	数据结构	C7	4
C6	数据处理		2
C7	PASCAL语言	C6	4

一、INSERT 语句（P100-101）

1、语法

- 插入一个元组

```
INSERT
INTO 〈表名〉 [（〈列名1〉[，〈列名2〉] …）]
VALUES （〈常量1〉[，〈常量2〉] …）；
```

- 插入子查询结果（可以是多个元组）

```
INSERT
INTO 〈表名〉[〈列名〉[，〈列名〉] …]
SELECT语句；
```

一、INSERT 语句

2、插入一个元组的实例

例1 插入一条选课记录（‘95020’, ‘C1’)

```
INSERT
INTO     SC (S# , C#)
VALUES   ( ‘95020’, ‘C1’) ;
```

*：属性G取空值。

例2 插入一条选课记录（‘95020’, ‘C1’, 80)

```
INSERT
INTO     SC
VALUES   ( ‘95020’, ‘C1’, 80) ;
```

*：INTO子句中没有指定列名。

一、INSERT 语句

3、插入子查询结果的实例

例3 对每个系，求学生的平均年龄，并把结果存入数据库

首先在数据库中建立一个新表

```
CREATE TABLE Deptage
        ( SD      CHAR(15),
          Avgage  SMALLINT );
```

将数据插入新表：

```
INSERT
INTO  Deptage (SD, Avgage)
              SELECT SD, AVG(SA)
              FROM  S
              GROUP BY SD;
```

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

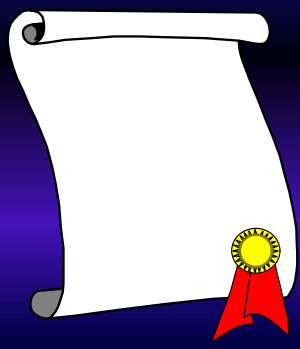
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



二、UPDATE 语句（P101-103）

1、语法

一条UPDATE语句在某一时刻只能更新一张表，但可以更新一张表中的多列，也可以更新多行数据。

UPDATE语句的语法形式为：

```
UPDATE 〈表名〉
SET 〈列名〉=〈表达式〉[, 〈列名=表达式〉]...
[WHERE 条件表达式];
```

语句中的WHERE子句决定需要更新的行，语句中的列名决定需要更新的列。

二、UPDATE 语句

2、修改一个元组的值

例4 将学生95001的年龄改为22岁。

```
UPDATE  S
SET SA = 22
WHERE S# = '95001';
```

二、UPDATE 语句

3、修改多个元组的值

例5 将所有学生的年龄增加1岁。

```
UPDATE   S
SET SA = SA + 1;
```

二、UPDATE 语句

4、带子查询的修改语句

子查询可以嵌套在UPDATE语句中，用于构造修改的条件。

例6 将计算机科学系全体学生的成绩置零。

```
UPDATE   SC
SET G = 0
WHERE 'CS' =
        ( SELECT SD
            FROM S
           WHERE S.S# = SC.S#
        ) ;
```

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

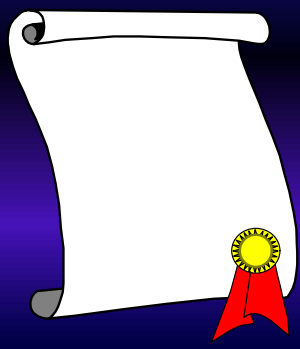
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



三、DELETE 语句

1、语法

语句DELETE用于从表中删除整行的数据，但不能用来从特定的列中删除数据。一条DELETE语句可以从表中删除一个单独的元组或是多个元组。

DELETE语句的语法形式为：

```
DELETE
FROM 〈表名〉
[WHERE 条件表达式];
```

三、DELETE 语句

2、删除一个元组

例7 删除学号为95001的学生记录。

```
DELETE
FROM   S
WHERE S# = '95001';
```

三、DELETE 语句

3、删除多个元组

例8 删除所有学生的选课记录。

```
DELETE
FROM   SC;
```

注意：这里删除的表中的数据，而不是表的定义，表的定义仍在DD中

三、DELETE 语句

4、带子查询的删除语句

子查询可以嵌套在DELETE语句中，用于构造执行删除操作的条件。

例9 删除计算机科学系全体学生的选课记录。

```
DELETE
FROM   SC
WHERE ‘CS’ =
        ( SELECT SD
          FROM S
          WHERE S.S# = SC.S#
        ) ;
```

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

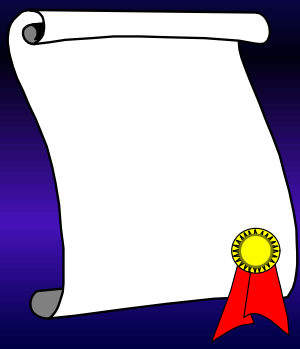
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



四、视图的定义（P104-107）

1、什么是视图？

- **视图是动态结果**：对一个或多个基本关系进行关系操作得到的动态结果
- **视图是虚关系**：一个无需存在于数据库中，但却可以根据某个特定用户的需求而生成的虚关系
- **视图的内容被定义成**基于一个或多个基本关系的**查询**
- **视图的操作**：视图可以和基本表一样被查询、被删除。也可以在一个视图之上再定义新的视图，但对视图的更新（增、删、改）操作则有一定的限制。

四、视图的定义

2、定义视图语法

视图可以从一张表、几张表或其他视图中创建。

```
CREATE VIEW〈视图名〉[ (〈列名〉[,〈列名〉] ...) ]
AS
SELECT 语句                    /*<子查询>*/
[WITH CHECK OPTION];
```

- ◆若视图名后的列名表与SELECT保留字后的列名表相同，则视图名后的列名表可以省略。
- ◆若使用WITH CHECK OPTION，则对视图进行UPDATE和INSERT操作时，保证更新时先满足视图定义中的WHERE子句指定的条件。

四、视图的定义

3、多种视图

① **水平视图**：从单个基本表导出，并且只是去掉了基本表的**某些行**

② **垂直视图**：从单个基本表导出，并且只是去掉了基本表的**某些列**

③ **分组视图**：用GROUP BY 子句定义的视图

④ **连接视图**：用**连接运算**得到的视图

四、视图的定义

4、实例 —— 创建水平视图

例10 创建一个视图，让分支机构B003的经理只看到他所在分支机构的职员的信息。

```
CREATE VIEW Manager3Staff
    AS      SELECT *
            FROM    Staff
            WHERE branchNo =  'B003' ;
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

四、视图的定义

4、实例 —— 创建垂直视图

例11 建立关于分支机构B003职员信息但不包括工资信息的视图。

```
CREATE VIEW Staff3
 AS SELECT staffNo, fName, lName, position, sex
    FROM     Staff
    WHERE  branchNo = ‘B003’ ;
```

或

```
CR
AS                                          on, sex
FR
```

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

四、视图的定义

4、实例 ——创建分组或连接视图

例12 创建管理出租房产的职员的视图，包括职员所在分支机构的编号、职员编号和他们管理的房产的数量。

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
    AS   SELECT   s.branchNo, s.staffNo, COUNT(*)
         FROM     Staff s, PropertyForRent p
         WHERE    s.staffNo = p.staffNo
         GROUP BY s.branchNo, s.staffNo;
```

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

四、视图的定义

5、WITH CHECK OPTION

- 迁移行：当对视图进行更新操作时，进入或离开视图的行称为迁移行
- WITH CHECK OPTION 子句禁止行迁移出视图。
- 如果指定WITH CHECK OPTION,当对视图进行INSERT/UPDATE操作时，有新的行不满足视图定义中的where条件（行迁移出视图），操作即被拒绝。

依次执行下列SQL语句后的结果是什么？

- INSERT INTO Manager3Staff(staffNo, branchNo) VALUE ('SG50', 'B002');
- UPDATE Manager3Staff SET branchNo = 'B007' WHERE staffNo = 'SG37';
- SELECT COUNT(*) FROM Staff;
- SELECT

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Manager3Staff视图定义如下：

```
CREATE VIEW Manager3Staff
   AS SELECT *
      FROM   Staff
      WHERE branchNo =  'B003' ;
```

Table 6.3 Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

WITH CHECT OPTION的使用

Manager3Staff视图的定义改为：

```
CREATE VIEW Manager3Staff
    AS SELECT *
       FROM    Staff
       WHERE branchNo =  'B003'
    WITH CHECK OPTION;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Table 6.3 Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

- INSERT INTO Manager3Staff(staffNo, branchNo) VALUE ('SG50', 'B002');

或者

- UPDATE Manager3Staff SET branchNo = 'B007' WHERE staffNo = 'SG37';

的执行结果是什么 ？

四、视图的定义

6、删除视图（见教材P107）

- 语法：

```
DROP VIEW ViewName [RESTRICT|CASCADE];
```

- 视图是虚表，删除的是视图的定义

  用DROP VIEW 语句删除视图时，只将数据字典中的视图的定义删除，对产生视图数据的基本表中的数据没有影响

- 默认RESTRICT ，如果存在依赖被删除视图的其他对象，则不允许删除

- 如果指定CASCADE，删除视图可能产生级联删除

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

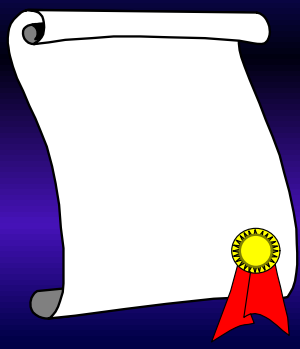
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



五、视图的查询（见教材P108-109）

- 视图定义后，用户可以象对基本表一样对视图进行查询
- 视图消解（View Resolution）---- 执行对视图的查询时，从数据字典中取出视图的定义，将定义中的子查询和用户的查询结合起来，转换成等价的对基本表的查询的过程
- 目前关系数据库系统对视图消解的支持有差异

五、视图的查询

例13 基于例12中创建的视图StaffPropCnt，计算在分公司B003工作的每位职员管理房产的数量。

```
SELECT    staffNo, cnt
FROM      StaffPropCnt
WHERE     branchNo = 'B003'
ORDER BY staffNo;
```

staffNo	cnt
SG14	1
SG37	2

```
CREATE VIEW       StaffPropCnt (branchNo, staffNo, cnt)
      AS  SELECT    s.branchNo, s.staffNo, COUNT(*)
          FROM      Staff s, PropertyForRent p
          WHERE     s.staffNo = p.staffNo
          GROUP BY s.branchNo, s.staffNo;
```

五、视图的查询

```
SELECT    staffNo, cnt
FROM      StaffPropCnt
WHERE     branchNo = 'B003'
ORDER BY staffNo;
```

```
CREATE VIEW       StaffPropCnt (branchNo, staffNo,
    cnt)
        AS  SELECT     s.branchNo, s.staffNo, COUNT(*)
            FROM       Staff s, PropertyForRent p
            WHERE      s.staffNo = p.staffNo
            GROUP BY s.branchNo, s.staffNo;
```

视图消解过程：

(1)将SELECT列表中给出的列名转换为视图定义中对应的列名：

SELECT s.staffNo AS staffNo, COUNT(*) AS cnt

(2)FROM子句的视图名以视图定义中的FROM列表代替：

FROM Staff s, PropertyForRent p

(3) 用AND将WHERE子句和视图定义中的WHERE子句合并：

WHERE s.staffNo = p.staffNo AND branchNo = 'B003'

(4)从视图定义中复制GROUP BY和HAVING子句

GROUP BY s.branchNo, s.staffNo

(5) 将 ORDER BY列名转换为视图定义中对应的列名

ORDER BY s.staffNo;

五、视图的查询

(6)最后，合并查询变为：

```
SELECT    s.staffNo AS staffNo, COUNT(*) AS cnt
FROM      Staff s, PropertyForRent p
WHERE     s.staffNo = p.staffNo AND branchNo =
'B003'
GROUP BY s.branchNo, s.staffNo
ORDER BY s.staffNo;
```

五、视图的查询

有些情况下，这种转换不能直接进行：

例13 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视图定义为：

```
CREATE VIEW  S_G (S# , Gavg)
  AS
    SELECT S# , AVG (G)
    FROM  SC
    GROUP BY S# ;
```

简单转换为：

```
SELECT S# , AVG（G）
FROM  SC
WHERE AVG(G) >= 90
GROUP BY S# ;
```

正确的转换应该是：

```
SELECT S# , AVG（G）
FROM  SC
GROUP BY S#
HAVING AVG(G) >= 90;
```

五、视图的查询

ISO标准对创建和使用视图施加了一些重要约束：

① 如果视图中某个列（如cnt）是基于集合函数的，那么，该列只能出现在访问视图的SELECT和ORDER BY子句中，特别是，该列不能出现在WHERE子句中，并且不能作为集合函数的参数：

例：
```
SELECT COUNT (cnt)
    FROM StaffPropCnt;
```

例：
```
SELECT *
    FROM StaffPropCnt
    WHERE cnt > 2;
```

② 分组视图不能与基本表或视图进行连接操作

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

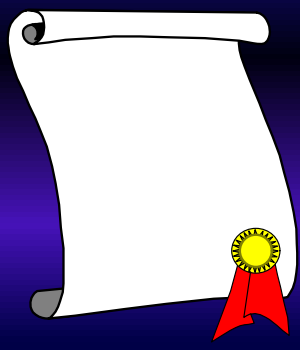
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



六、视图的更新

- 对视图操作的语法与对基本表的一样
- 对基本表的所有更新应该立即反映到涉及该表的所有视图中；同样地，如果一个视图被更新，该视图涉及到的基本表也应该能反映这种变化。
- 视图都可以更新吗？

并不是所有的视图都是可更新的，因为有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（教材P110）

六、视图的更新

例14 将分支机构B003的职员SG5管理两处房产的记录插入视图StaffPropCnt:

```
INSERT INTO StaffPropCnt
    VALUES      ('B003', 'SG5', 2);
```

Table 6.5 Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

?

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

六、视图的更新

例15 如下建立视图StaffPropList

```
CREATE VIEW StaffPropList (branchNo, staffNo, propertyNo)
    AS SELECT  branchNo, staffNo, propertyNo
        FROM PropertyForRent;
```

插入一行

```
INSERT INTO StaffPropList
VALUES        ('B003', 'SG5', 'PG19');
```

StaffPropList

propertyNo	staffNo	branchNo
PA14	SA9	B007
PL94	SL41	B005
PG4		B003
PG36	SG37	B003
PG21	SG37	B003
PG16	SG14	B003

?

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

六、视图的更新

```
CREATE TABLE PropertyForRent (
    propertyNo    PNumber          NOT NULL, ….
    rooms         PRooms           NOT NULL      DEFAULT 4,
    rent          PRent            NOT NULL,     DEFAULT 600,
    ownerNo       OwnerNumber      NOT NULL,
    staffNo       StaffNumber      Constraint StaffNotHandlingTooMuch ….
    branchNo      BranchNumber     NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff
      ON DELETE SET NULL ON UPDATE CASCADE ….);
```

表PropertyForRent的定义中，除了staffNo外所有列都不允许为空值。但是，视图StaffPropList中并不包括表中除了propertyNo以外的其它列，因此，无法提供非空的列值

六、视图的更新

例16 将信息系学生视图IS_S中学号为‘95002’的学生姓名改为‘刘辰’。

```
UPDATE   IS_S
SET   SN = ‘刘辰’
WHERE S# = ‘95002’;
```

IS_S视图定义为：

```
CREATE VIEW   IS_S
       AS
       SELECT S# , SN, SA
       FROM   S
       WHERE SD= ‘IS’ ;
```

视图消解后的更新语句为：

```
UPDATE   S
SET    SN =  ‘刘辰’
WHERE   S# =  ‘95002’
   AND   SD =  ‘IS’ ;
```

六、视图的更新

有些更新不能有意义地转换成对基本表S的更新：

例17 将视图S_G中学号‘95001’的学生的平均成绩改为90分。

```
UPDATE  S_G
SET Gavg = 90
 WHERE S# = ‘95001’;
```

S_G视图定义为：

```
CREATE VIEW  S_G (S# , Gavg)
        AS
        SELECT S# , AVG (G)
        FROM  SC
        GROUP BY S# ;
```

六、视图的更新

- ISO标准给出的视图可更新的充要条件为：

① 没有指定DISTINCT，即重复元组未从查询结果中消除；

② 定义查询的SELECT列表中的每个元素均为列名（而不是常量，表达式或聚合函数），且列名不能出现多于一次；

③ FROM子句只能指定一个表，即视图必须有一个源表且用户有请求该表的权限。如果源表本身就是一个视图，那么视图必须满足这些条件。因此，排除了基于连接、并（UNION）、交（INTERSECT）或差（EXCEPT）操作的所有视图；

④ WHERE子句不能包括任何引用了FROM子句中的表的嵌套SELECT操作；

⑤ 定义查询不能有GROUP BY或HAVING子句。

另外，添加到视图中的每一行都不能违反基本表的完整性约束

六、视图的更新

- **可更新视图**---- 视图的更新能唯一地有意义地转换成对基本表的更新时，该视图是可更新的
- 一般地，**行列子集视图**是理论上可更新的
  - 行列子集视图 ---- 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列，但保留了键，则称这类视图为行列子集视图。
- **不可更新的视图**（理论上不可更新的视图）---- 当有些视图的更新不能唯一地有意义地转换成对基本表的更新时，该视图是不可更新的
- **不允许更新的视图**（各关系数据库系统对视图更新的支持程度不同）

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

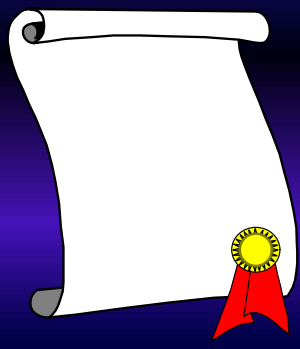
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



七、视图的优缺点

1、视图的优点（参考教材P111-112）

- 数据独立性
- 实时性
- 提高了安全性
- 降低了复杂性
- 使用户从多种角度看待同一数据
- 数据完整性

七、视图的优缺点

2、视图的缺点

- 更新的局限性
- 结构的局限性
- 性能开销

七、视图的优缺点

3、视图物化（View Materialization）

- 视图维护：更新基本表引起视图更新的过程称为视图维护
- 视图物化：把第一次访问视图的结果存储为数据库的临时表；这样，基于物化视图的查询比每次重新计算视图要快的多。

本讲主要内容

一. INSERT语句

二. UPDATE语句

三. DELETE语句

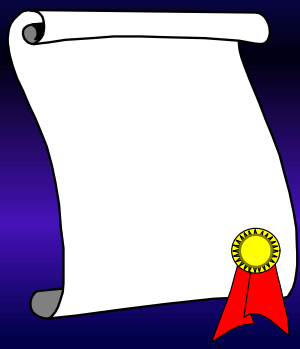
四. 视图的定义

五. 视图的查询

六. 视图的更新

七. 视图的优缺点

八. SQL的完整性约束



八、SQL的完整性约束

（参考教材P149-159）

概念：数据库的完整性是指数据库中数据在逻辑上的正确性、有效性和相容性。

功能：防止数据库中存在不符合语义的数据，防止错误信息的输入和输出，即所谓**垃圾进垃圾出**所造成的无效操作和错误结果。

八、SQL的完整性约束

SQL中五种类型的非过程性完整性约束

① 数据取值要求

② 域约束

③ 实体完整性

④ 参照完整性

⑤ 一般约束（企业约束）

SQL中的过程性完整性约束

⑥ 触发器

八、SQL的完整性约束

1、数据取值要求

- 非空约束

例如：position VARCHAR(10) NOT NULL

- 唯一性约束

例如：SN CHAR（8） UNIQUE

- CHECK约束 限制列中值的范围

格式： **CHECK**(searchCondition)

例如： sex CHAR NOT NULL CHECK (sex IN ('M', 'F'))

八、SQL的完整性约束

2、域约束

CREATE DOMAIN 创建域，域本质上是一种可带约束的数据类型. 域是一组具有相同数据类型的值的集合。

（1）创建域

语法：

```
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
```

例如：

```
CREATE DOMAIN SexType AS CHAR
        CHECK (VALUE IN ('M', 'F'));
sex   SexType   NOT NULL
```

八、SQL的完整性约束

说明：

searchCondition 可以用SELECT语句表达：

例如： **CREATE DOMAIN** BranchNo **AS CHAR**(4)
CHECK (**VALUE IN** (SELECT branchNo
FROM Branch));

(2) 删除域

使用DROP DOMAIN删除域：

语法：

```
DROP DOMAIN DomainName [RESTRICT | CASCADE]
```

例如： DROP DOMAIN SexType;

八、SQL的完整性约束

3、实体完整性

- 定义主键

PRIMARY KEY （<列名>[，<列名>] …）]

- 关系R中任意行在主键列的取值都不允许为空

- 关系系统自动支持实体完整性；
- 可能破坏实体完整性规则的操作
  - 插入和更新

八、SQL的完整性约束

4、参照完整性

- 定义外键

FOREIGN KEY （<列名>[，<列名>] …）
REFERENCES <被参照表名>] ）；

- 关系系统自动支持参照完整性；
- 增删改操作一次只能对一个表进行操作；更新操作可能破坏关系规则，从而破坏数据库的一致性；
- 破坏参照完整性的操作有：
  - 对参照表(子表)的插入和更新操作
  - 对被参照表(父表)的删除和更新操作

一个学生-课程数据库

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

C

课程号 C#	课程名 CN	先行课 CP#	学分 CC
C1	数据库	C5	4
C2	数学		2
C3	信息系统	C1	4
C4	操作系统	C6	3
C5	数据结构	C7	4
C6	数据处理		2
C7	PASCAL语言	C6	4

SC

学号 S#	课程号 C#	成绩 G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

*回忆一下UNIT5中这三个表的定义语句

八、SQL的完整性约束

操作一：删除学号为“95002”的学生的信息；

解决方法一：

```
DELETE
FROM   S
WHERE  S# = '9500...

DELETE
FROM   SC
WHERE  S# = '9500...;
```

破坏参照完整性

解决方法二：

```
DELETE
FROM   SC
WHERE  S# = '95002' ;

DELETE
FROM   S
WHERE  S# = '95002' ;
```

两个SQL删除命令的顺序不合适会破坏参照完整性

八、SQL的完整性约束

操作二：插入选课信息（‘95005 ‘C8’ ,85）;

```
INSERT INTO S
VALUES ( ‘95005’ , null, null,null,null) ;

INSERT INTO C
VALUES ( ‘C8’ , null, null) ;

INSERT INTO SC
VALUES ( ‘95005 ‘C8’ ,85) ;
```

三个SQL插入命令的顺序不合适会破坏参照完整性

八、SQL的完整性约束

操作三：修改学号“95001”的学生的学号为“95008”；

```
UPDATE S
SET S# = '95008'
WHERE S# = '95001';

UPDATE SC
SET S# = '95008'
WHERE S# = '95001';
```

破坏参照完整性

```
UPDATE SC
SET S# = '95008'
WHERE S# = '95001';

UPDATE S
SET S# = '95008'
WHERE S# = '95001';
```

破坏参照完整性

八、SQL的完整性约束

参照完整性被破坏时可选择的策略:

DBMS在实现参照完整性时，除了要提供定义主键、外键的机制外，还需要提供不同的策略供用户选择。根据应用环境的要求，选择策略(Full SQL-99)：

- NO ACTION
- CASCADE
- SET NULL
- SET DEFAULT
- RESTRICT

八、SQL的完整性约束

- 在父表中用DELETE或UPDATE操作删除父表与子表有匹配行的键值时
  - --- CASCADE：删除父表中的行，且删除子表中匹配的行
  - --- SET DEFAULT：删除父表中的行，且自动设置子表中的外键值为缺省值
  - --- SET NULL：删除父表中的行，且自动设置子表中的外键值为NULL
  - --- NO ACTION / RESTRICT 拒绝对父表进行的操作。是默认的动作
- 在子表中用INSERT或UPDATE操作插入与父表中键值不匹配的外键值时
  - --- RESTRICT：拒绝对子表进行的操作

八、SQL的完整性约束

定义表语句的语法扩充：

```
CREATE TABLE 〈表名〉
  (  (〈列名〉〈数据类型〉[NOT NULL | NULL] [UNIQUE] [DEFAULT 缺省值]
     [, 〈列名〉〈数据类型〉[NOT NULL | NULL] [UNIQUE]] [DEFAULT 缺省值] ...
     [, PRIMARY KEY  (〈列名〉[, 〈列名〉] ...) ]
     FOREIGN KEY  (〈列名〉[, 〈列名〉] ...) REFERENCES 〈被参照表名〉
     [ON DELETE [CASCADE | SET DEFAULT | SET NULL | NO ACTION] ]
     [ON UPDATE [CASCADE | SET DEFAULT | SET NULL |NO ACTION] ]
  ) ;
```

不同的实现对该版本的支持有区别

八、SQL的完整性约束

定义参照完整性约束时，分别对DELETE和UPDATE指定约束违背时的自动触发动作：

```
FOREIGN KEY  (〈列名〉[, 〈列名〉] ...)
REFERENCES    〈被参照表名〉
[ON DELETE [CASCADE | SET DEFAULT | SET NULL
| NO ACTION] ]
[ON UPDATE [CASCADE | SET DEFAULT | SET NULL
|NO ACTION] ]
```

使用SET DEFAULT选项的前提是：在创建子表时对外键属性指定了DEFAULT值

八、SQL的完整性约束

5、一般约束

- 在语句CREATE and ALTER TABLE中使用CHECK/UNIQUE
- 与CHECK子句类似，有：

CREATE ASSERTION AssertionName
CHECK (searchCondition)

- 例如：

CREATE ASSERTION StaffNotHandlingTooMuch
CHECK (NOT EXISTS (SELECT *
FROM PropertyForRent
GROUP BY staffNo
HAVING COUNT(*) > 100))

*回忆一下UNIT5中PropertyForRent表的定义语句

八、SQL的完整性约束

5. 事务（Transaction）

- 事务

  ----- 是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。

- SQL定义事务的语句

  BEGIN TRANSACTION

  COMMIT

  ROLLBACK

八、SQL的完整性约束

5. 事务（Transaction）

- 约束的监测时机可以是事务结束时或者每一条SQL语句执行完成时；
- SET CONSTRAINTS 语句用于指定约束检测的时机；
- 语句格式：

SET CONSTRAINTS

{ALL | constraintName [,…]} {DEFERRED | IMMEDIATE}

- IMMEDIATE是缺省值

八、SQL的完整性约束

6、触发器

触发器（Trigger）就是一类靠事件驱动的特殊过程，一旦由某个用户定义，任何触发该触发器的事件发生时，均由服务器自动激活相应的触发器。

- 触发器包含三个要素：
  - 触发事件（条件）
  - 触发时机
  - 触发动作

八、SQL的完整性约束

- 触发器的定义
  - 指明什么条件下触发器被执行
  - 指明触发器执行的动作是什么
- 触发器的作用
  - 示警
  - 满足特定条件时自动执行某项任务
- 触发器事件

  Insert、 delete、update

八、SQL的完整性约束

◆定义触发器的语句(Full SQL-99的Create Trigger语法)

```
CREATE TRIGGER trigger_name {BEFORE | AFTER}
{INSERT | DELETE | UPDATE [OF columnname{,columnname…}]}
ON tablename [REFERENCING corr_name_def {, corr_name_def…}]
[FOR EACH ROW | FOR EACH STATEMENT]
        [WHEN (search_condition)]
        {statement
         |BEGIN ATOMIC statement; {statement; …} END} ;
```

定义行的相关名字的corr_name_def 如下:

```
{OLD [ROW] [AS] old_row_corr_name
 | NEW [ROW] [AS] new_row_corr_name
 | OLD TABLE [AS] old_table_corr_name
 | NEW TABLE [AS] new_table_corr_name}
```

可读性好些的写法:

create trigger trigger-name { before | after }

{ insert | delete | update [of column-name] }

on table-name

referencing { old row as identifier | new row as identifier }

for { each row | each statement }

when(search-condition)

{ statement | begin atomic triggered-SQL-statement end }

八、SQL的完整性约束

◆用触发器实现过程性约束

- 在定义触发器动作时,各DBMS产品用自己特定的过程性语言,不可能给出一个Create Trigger的”基本SQL”版本
- 不同产品的Create Trigger形式中,非动作部分很相似;
- 大部分DBMS,如SYSBASE、ORACLE和INFORMAX都提供包含常驻内存的变量、循环控制和if-then-else逻辑的过程语言的扩展.如SYSBASE、ORACLE和INFORMAX的T-SQL 、PL/SQL和SPL;DB2 UDB的过程扩展性少一些

八、SQL的完整性约束

◆用触发器实现完整性约束

例,使用触发器来检查,在学生表中,新的学生行中,性别必须是’F’或’M’

```
create  trigger sse  after insert  on s
   referencing new as x
   for each row when x .se not in (‘F’, ‘M’)
   begin
   raise_application_error(-20003, ‘invalid discount attempted on insert’);
   end ;
```

八、SQL的完整性约束

◆在MYSQL中,用触发器实现完整性约束

例,使用触发器来保证在学生表中删除一个学生的记录，则在选课表中该生的相应记录也被删除。

```
create trigger tsdsc
 after delete on s
 for each row
 delete from sc where sc.sno=old.sno;
```

八、SQL的完整性约束

- EMP(ENO, ENAME, SAL, JOB)

职工工资增幅不得超过10%

```
create trigger RAISE_LIMIT
after update of SAL on EMP
        referencing new row as nrow old row as orow
        for each row
        when (nrow.SAL > 1.1 * orow.SAL)
            begin
             raise_application_error (-20005, 'Salary is
        increased  more than 10%')
            end;
```

八、SQL的完整性约束

当帐户透支时，将帐户余额设为0，并建一笔贷款，其金额为透支额

```
create trigger overdraft-trigger after update on account
        referencing new row as nrow  for each row
        when nrow.balance < 0
        begin
            insert into borrower
                        (select customer-name, account-number
                        from depositor
                        where nrow.account-number = depositor.account-number);
                        insert into loan values(nrow.account-number,
                                                nrow.branch-name, – nrow.balance);
        update account set balance = 0
            where account.account-number = nrow.account-number
        end
```

八、SQL的完整性约束

◆非过程性约束与过程性约束的优缺点

- Create Table语句中的非过程性约束的种类是有限的(CHECK约束,UNIQUE约束,PRIMARY KEY约束,FOREIGN KEY约束),因而能力是有限的,但容易理解;
- 非过程性约束直接为系统所知;
- 非过程性约束难以给出约束不满足时的相应动作;
- 过程性约束的作用更大
  - ☆ 给出约束不满足时的相应动作
  - ☆ 保证事务的一致性(可以使一个更新操作在另一个执行之后自动发生)

Questions?

本讲主要目标

学完本讲后，你应该能够了解：

1、三个SQL更新语句的使用；

2、视图是虚表，视图在定义时，并没有真正执行定义语句中的查询语句；用户可以像使用基本表一样对视图进行查询和更新操作，但有时候是有限制的；

3、视图的作用；

4、SQL的更新操作可能破坏数据库的一致性；SQL提供了完整性约束的定义功能。

问题讨论

1、SQL的三种更新操作的粒度分别是什么？

2、视图的更新是否会破坏完整性约束？

练习

教材：《数据库系统原理教程》（第2版）

P112

1）5

2）6