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 数据定义
3.3 查询
3.4 数据更新
3.5 视图
3.6 小结

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 删除数据

1. 插入单个元组

- 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。

插入单个元组（续）

[例] 将一个新学生记录（学号：95020；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

```
INSERT  INTO Student
VALUES ('95020'，'陈冬'，'男'，'IS'，18)；
```

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;

- 功能

  将子查询结果插入指定表中

插入子查询结果（续）

[例] 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  Deptage
   (Sdept  CHAR(15)          /* 系名*/
   Avgage SMALLINT);    /*学生平均年龄*/
```

插入子查询结果（续）

第二步：插入数据

```
INSERT
INTO  Deptage(Sdept，Avgage)
    SELECT  Sdept，AVG(Sage)
    FROM  Student
    GROUP BY Sdept;
```

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 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

1. 修改某一个元组的值

[例] 将学生95001的年龄改为22岁。

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 95001 ';
```

2. 修改多个元组的值

[例] 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] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC
SET  Grade=o
WHERE  'CS'=
     (SELETE Sdept
      FROM  Student
      WHERE  Student.Sno = SC.Sno);
```

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 删除数据

3.4.3 删除数据

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
  - 缺省表示要修改表中的所有元组

1. 删除某一个元组的值

[例] 删除学号为95019的学生记录。

```
DELETE
FROM Student
WHERE Sno='95019';
```

2. 删除多个元组的值

[例] 删除2号课程的所有选课记录。

```
DELETE
FROM SC;
WHERE Cno='2';
```

[例] 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] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM SC
WHERE  'CS'=
    (SELETE Sdept
     FROM Student
     WHERE Student.Sno=SC.Sno);
```

第三章 关系数据库标准语言SQL

3.1 SQL概述
3.2 数据定义
3.3 查询
3.4 数据更新
3.5 视图
3.6 小结

3.5 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，不会出现数据冗余
- 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

3.5 视　图

基于视图的操作

- 定义
- 查询
- 删除
- 受限更新
- 定义基于该视图的新视图

3.5 视　图

3.5.1 定义视图

3.5.2 查询视图

3.5.3 更新视图

3.5.4 视图的作用

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];

行列子集视图

[例] 建立信息系学生的视图。

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

从单个基本表导出,只是去掉了基本表的某些行和某些列,保留了码————行列子集视图

建立视图（续）

- WITH CHECK OPTION

透过视图进行增删改操作时，不得破坏视图定义中的谓词条件（即子查询中的条件表达式）

基于多个基表的视图

[例] 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';
```

基于视图的视图

[例] 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;
```

带表达式的视图

[例] 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，2020-Sage
FROM  Student
```

设置一些派生属性列，也称为虚拟列--Sbirth 带表达式的视图必须明确定义组成视图的各个属性列名

建立分组视图

[例] 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;
```

2. 删除视图

- DROP VIEW <视图名>;
  - 该语句从数据字典中删除指定的视图定义
  - 由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须显式删除
  - 删除基表时，由该基表导出的所有视图定义都必须显式删除

删除视图(续)

[例] 删除视图IS_S1

```
DROP VIEW IS_S1;
```

3.5 视　图

3.5.1 定义视图

3.5.2 查询视图

3.5.3 更新视图

3.5.4 视图的作用

3.5.2 查询视图

- 从用户角度：查询视图与查询基本表相同
- DBMS实现视图查询的方法
  - 实体化视图（View Materialization）
    - 有效性检查：检查所查询的视图是否存在
    - 执行视图定义，将视图临时实体化，生成临时表
    - 查询视图转换为查询临时表
    - 查询完毕删除被实体化的视图(临时表)

查询视图（续）

[例] 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;
```

IS_Student视图的定义 (视图定义例1)：

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

查询视图（续）

[例] 查询信息系选修了1号课程的学生

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

3.5 视　图

3.5.1 定义视图

3.5.2 查询视图

3.5.3 更新视图

3.5.4 视图的作用

3.5.3 更新视图

- 用户角度：更新视图与更新基本表相同
- DBMS实现视图更新的方法
  - 视图实体化法（View Materialization）
  - 视图消解法（View Resolution）
- 指定WITH CHECK OPTION子句后

DBMS在更新视图时会进行检查，防止用户通过视图对不属于视图范围内的基本表数据进行更新

更新视图（续）

[例] 将信息系学生视图IS_Student中学号95002 的学生姓名改为“刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= '95002';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= '95002' AND Sdept= 'IS';
```

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(对两类方法均如此)

例：视图S_G为不可更新视图。

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

更新视图（续）

对于如下更新语句：

```
UPDATE  S_G
SET     Gavg=90
WHERE  Sno= '95001';
```

无论实体化法还是消解法都无法将其转换成对基本表SC的更新

实际系统对视图更新的限制

- 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

DB2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。

更新视图（续）

(3) 若视图的字段来自集函数，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。

(7) 一个不允许更新的视图上定义的视图也不允许更新

3.5 视　图

3.5.1 定义视图

3.5.2 查询视图

3.5.3 更新视图

3.5.4 视图的作用

1. 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

2. 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

3. 视图对重构数据库提供了一定程度的逻辑独立性

- 物理独立性与逻辑独立性的概念
- 视图在一定程度上保证了数据的逻辑独立性
- 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。

4. 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
- 通过WITH CHECK OPTION对关键数据定义操作时间限制

建立视图（续）

[例] 建立1号课程的选课视图，并要求透过该视图进行的更新操作只涉及1号课程，同时对该视图的任何操作只能在工作时间进行。

```
CREATE VIEW IS_SC
            AS
SELECT Sno，Cno，Grade
FROM  SC
WHERE  Cno= '1'
AND  TO_CHAR(SYSDATE,'HH24')  BETWEEN  9  AND  17
AND  TO_CHAR(SYSDATE,'D')  BETWEEN  2  AND  6
WITH CHECK OPTION;
```

第三章 关系数据库标准语言SQL

3.1 SQL概述
3.2 数据定义
3.3 查询
3.4 数据更新
3.5 视图
3.6 小结

本章总结

- 主要内容：
  SQL概述，数据定义，查询，更新，视图，数据控制
- 重要知识点：
  SQL语句的编写
- 本章题型：
  选择，填空，应用题

本章习题

本章习题5

1.SELECT SNAME,CITY
FROM S;

2.SELECT PNAME,COLOR,WEIGHT
FROM P;

3.SELECT DISTINCT(JNO)
FROM SPJ
WHERE SNO='S1';

4.SELECT P.PNAME,SPJ.QTY
FROM P,SPJ
WHERE P.PNO=SPJ.PNO AND SPJ.JNO='J2';

```
5.SELECT DISTINCT PNO
   FROM SPJ
   WHERE SNO IN  (SELECT SNO
                  FROM S
                  WHERE CITY=‘上海’);
6.SELECT JNAME
   FROM J,SPJ,S
   WHERE     J.JNO=SPJ.JNO AND
             SPJ.PNO=S.PNO AND
             S.CITY=‘上海’;
7.SELECT JNO
   FROM SPJ
   WHERE NOT EXISTS (SELECT * FROM S
                     WHERE SPJ.SNO=S.SNO
                           AND S.CITY=‘天津’);
```

8.UPDATE P
SET COLOR=‘蓝’
WHERE COLOR=‘红’ ;

9.UPDATE SPJ
SET SNO=‘S3’
WHERE SNO=‘S5’ AND
JNO=‘J4’ AND PNO=‘P6’;

10.DELETE
FROM SPJ
WHERE SNO=‘S2’;
DELETE
FROM S
WHERE SNO=‘S2’;

11.INSERT INTO SPJ
VALUES(S2,P4,J6,200);

习题11

```
CREATE VIEW SPJ_SANJIAN
AS      SELECT SNO，PNO，QTY
        FROM SPJ
        WHERE JNO=（SELECT JNO
                    FROM J
                    WHERE JNAME=‘三建’）；
```

1.SELECT PNO,QTY
FROM SPJ_SANJIAN;

2.SELECT PNO,QTY
FROM SPJ_SANJIAN
WHERE SNO=‘S1’;

习题4

```
1.SELECT SNO        FROM SPJ
   WHERE JNO='J1';
2.SELECT SNO        FROM SPJ
   WHERE JNO='J1' AND PNO='P1';
3.SELECT SNO        FROM SPJ,P
   WHERE JNO='J1' AND     SPJ.PNO=P.PNO AND  COLOR='红' ;
4.SELECTJNO
   FROM J
   WHERE NOT EXISTS
    ( SELECT *  FROM SPJ,S,P
    WHERE SPJ.JNO=J.JNO    AND
              SPJ.SNO=S.SNO AND  SPJ.PNO=P.PNO  AND
              S.CITY='天津'  AND  P.COLOR='红' );
```

1.查询所有比‘李三’年龄大的学生姓名，年龄和性别。

```
SELECT SNO,SAGE,SSEX   FROM S
WHERE SAGE>(SELECT SAGE FROM S
                    WHERE SNAME = ‘李三’);
```

2.查询选修课程号为2的学生中成绩最高的学生号。

```
SELECT SNO FROM SC
WHERE CNO = ‘2’ AND
        GRADE>=ALL(SELECT GRADE FORM SC
                    WHERE CNO=‘2’);
```

3.查询选修四门以上课程的总成绩（不统计不及格的课程），并要求按总成绩的降序排列。

```
SELECT SNO,SUM(GRADE) FROM SC
WHERE GRADE>=60
GROUP BY SNO HAVING COUNT(*)>=4
ORDER BY 2 DESC;
```

4.已知三个关系R(A,B,C),S(A,D,E)和T(D,F)，其中属性C和E为数值型，试用SQL语句实现：

- 将R，S和T三个关系按关联属性建立一个视图R-S-T；

```
CREATE VIEW R-S-T
AS SELECT R.A,B,C,S.D,E,F      FROM R,S,T
WHERE R.A=S.A AND S.D=T.D;
```

- 对视图按属性A分组后，求属性C和E的平均值；

```
SELECT A,AVG(C),AVG(E)     FROM R-S-T
GROUP BY A;
```

5.设关系R和S试用SQL语句实现：

- 查询属性C>50时，R中与S相关联的属性B的值；

```
SELECT B
FROM R,S
WHERE R.A=S.A AND C>50;
```

- 当属性C=40时，将R中与S相关联的属性B值修改为b4。

```
UPDATE R
SET R.B='b4'
WHERE A IN (SELECT A
                FROM S
                WHERE C=40);
```