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

3.4 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

3.4.1 插入数据

- 两种插入数据方式
  - 插入单个元组
  - 插入子查询结果

1. 插入单个元组

- 语句格式

INSERT INTO <表名> [(<属

[，<属性列2 >…)]

VALUES (<常量1> [，<常量

- 功能

将新元组插入指定表中。

INTO子句

1、指定要插入数据的表名及属性列

2、属性列的顺序可与表定义中的顺序不一致

3、**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

4、**指定部分属性列**：插入的元组在其余属性列上取空值

VALUES子句

提供的值必须与**INTO**子句匹配

1、值的个数

2、值的类型

插入单个元组（续）

[例1] 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)；
```

插入单个元组（续）

[例2] 插入一条选课记录('95020'，'1 ')。

```
INSERT INTO SC(Sno，Cno)
VALUES (' 95020 '，' 1 ');
```

新插入的记录在Grade列上取空值

2. 插入子查询结果

- 语句格式

  INSERT INTO <表名> [(<属性

  子查询;

- 功能

  将子查询结果插入指定表中

INTO子句

1、指定要插入数据的表名及属性列

2、属性列的顺序可与表定义中的顺序不一致

3、**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

4、**指定部分属性列**：插入的元组在其余属性列上取空值

VALUES子句

SELECT子句目标列必须与INTO子句匹配

1、值的个数

2、值的类型

插入子查询结果（续）

[例3] 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；
```

插入子查询结果（续）

DBMS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

- 对于有**NOT NULL**约束的属性列是否提供了非空值
- 对于有**UNIQUE**约束的属性列是否提供了非重复值
- 对于有值域约束的属性列所提供的属性值是否在值域范围内

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>[,
  [WHERE <条件>];

- 功能

修改指定表中满足WHERE子

SET子句

指定修改方式

要修改的列

修改后取值

WHERE子句

指定要修改的元组

缺省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4] 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5] 将所有学生的年龄增加1岁。

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

修改多个元组的值(续)

[例6] 将信息系所有学生的年龄增加1岁。

```
UPDATE Student
SET Sage= Sage+1
WHERE Sdept=' IS ';
```

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

修改数据（续）

DBMS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

3.4 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

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8] 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9] 删除2号课程的所有选课记录。

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

[例10] 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11] 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)；
```

删除数据(续)

DBMS在执行删除语句时会检查所删除的元组是否破坏表上已定义的完整性规则

- 参照完整性
  - 不允许删除
  - 级联删除

更新数据与数据一致性

DBMS在执行插入、删除、更新语句时必须保证数据库一致性

- 必须有事务的概念和原子性
- 完整性检查和保证

第三章 关系数据库标准语言SQL

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

- 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

  >]...)]

  AS <子查询>

  [WITH CHECK OPTIO

全部省略或全部指定

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:

(1) 某个目标列是集函数或列表达式

(2) 目标列为 *

(3) 多表连接时选出了几个同名列作为视图的字段

(4) 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）

DBMS执行CREATE VIEW语句时只是把视图的定义存入数据字典，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。

行列子集视图

[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'；
```

视图从单个基本表导出，并且只是去掉了基本表的某些行和某些列，但保留了码（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）

WITH CHECK OPTION的视图

[例2] 建立信息系学生的视图，并要求透过该视图进行的更新操作只涉及信息系学生。

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

对IS_Student视图的更新操作

- **修改操作：DBMS自动加上Sdept= 'IS'的条件**
- **删除操作：DBMS自动加上Sdept= 'IS'的条件**
- **插入操作：DBMS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'**

基于多个基表的视图

[例4] 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5] 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6] 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，2019-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7] 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）

- 一类不易扩充的视图
  - 以 SELECT * 方式创建的视图可扩充性差，应尽可能避免

建立视图（续）

[例8]将Student表中所有女生记录定义一个视图。

```
CREATE VIEW
 F_Student1(stdnum，name，sex，age，dept)
 AS
   SELECT *
    FROM  Student
    WHERE Ssex='女'；
```

缺点：修改基表Student的结构后，Student表F_Student1视图的映象关系被破坏，导致该视图不能正确工作。

建立视图（续）

```
CREATE VIEW
F_Student2 (stdnum，name，sex，age，dept)
 AS
      SELECT Sno，Sname，Ssex，Sage，Sdept
       FROM  Student
        WHERE Ssex='女'；
```

为基表Student增加属性列不会破坏Student表与F_Student2视图的映象关系。

常见的视图形式

- 行列子集视图
- **WITH CHECK OPTION**的视图
- 基于多个基表的视图
- 基于视图的视图
- 带表达式的视图
- 分组视图

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9] 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）

- 视图消解法（View Resolution）
  - 进行有效性检查，检查查询的表、视图等是否存在。如果存在，则从数据字典中取出视图的定义
  - 把视图定义中的子查询与用户的查询结合起来，转换成等价的对基本表的查询
  - 执行修正后的查询

[例1] 在信息系学生的视图中找出

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

IS_Student视图的定义 (视图定

```
CREATE VIEW IS_Stude
 AS
      SELECT Sno，Snam
      FROM  Student
      WHERE  Sdept= 'IS'；
```

1、视图实体化法

2、视图消解法

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2] 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'；
```

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查询。采用视图消解法的DBMS会限制这类查询。

查询视图（续）

[例3]在S_G视图中查询平均成绩
生学号和平均成绩。

```
SELECT *
FROM   S_G
WHERE  Gavg>=90;
```

S_G视图定义：

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

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;
```

3.5 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

[例1] 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（续）

[例2] 向信息系学生视图IS_Student中插入一个新的学生记录：95029，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('95029'，'赵新'，20，'IS' )；
```

更新视图（续）

[例3] 删除视图IS_Student中学号为95029的记录。

```
DELETE
FROM IS_Student
WHERE Sno= '95029';
```

转换为对基本表的更新：

```
DELETE
FROM Student
WHERE Sno= '95029' 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

视图的可更新性

理论上可更新

理论上不可更新

允许更新

不允许更新

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

例：数据库逻辑结构发生改变

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)

3.视图对重构数据库提供了一定程度的逻辑独立性

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，从而对原Student表的查询程序不必修改

3. 视图对重构数据库提供了一定程度的逻辑独立性

- 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3] 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;
```

练习：

- 请为三建工程项目建立一个供应情况的视图，包括供应商代码(SNO)、零件代码(PNO)、供应数量(QTY)。针对该视图完成下列查询：
  - 找出三建工程项目使用的各种零件代码及数量。
  - 找出供应商S1的供应情况。

建视图：

```
CREATE  VIEW  V_SPJ
  AS
     SELECT  SNO, PNO, QTY
     FROM  SPJ
     WHERE  JNO=
              ( SELECT  JNO
                FROM  J
                WHERE  JNAME='三建'  )
```

1、找出三建工程项目使用的各种零件代码及数量。

```
SELECT  PNO, QTY
FROM  V_SPJ
```

2、找出供应商S1的供应情况。

```
SELECT  PNO, QTY
FROM  V_SPJ
WHERE  SNO='S1'
```