第三章 关系数据库标准语言SQL

Principles of Database Systems

第三章　关系数据库标准语言SQL

3.5 数据更新

3.5.1 插入数据

插入数据是把新的记录插入到一个存在的表中。

1. 元组值的插入

语法：

```
INSERT INTO 基本表[(列名[, 列名]…)] VALUES(元组值);
```

作用：将一条元组值插入到表中。

注意：

- 列名的排列顺序不一定和表定义时的顺序一致。
- INTO子句中指定列名时，元组值的字段排列顺序必须和列名的排列顺序一致、个数相等、数据类型匹配。
- INTO子句中未指定列名时，元组值必须在每个属性列上均有值，且值的排列顺序与表中属性列的排列顺序一致。

3.5.1 插入数据

例1：往基本表SC中插入一个元组值

INSERT INTO SC VALUES('S004','C011',90);

例2：往基本表SC中插入部分元组值

INSERT INTO SC(sno, cno) VALUES('S004','C025');

注：对于INTO子句中没有出现的列，则新插入的记录在这些列上将取空值或缺省值，如上例的grade字段即赋空值。但有NOT NULL约束的属性列不能取空值。

3.5.1 插入数据

2. 查询结果的插入

将一个表中的数据抽取若干行插入另一表中，可以通过子查询来实现。

语法：

```
INSERT INTO 基本表名[(列名表)] 子查询;
```

作用：将子查询返回的结果数据集插入到表中。其功能是批量插入。

要求：查询语句的目标列必须与INTO子句匹配。

3.5.1 插入数据

例：将信息系（IS）平均成绩大于80分的学生学号和平均成绩存入另一个基本表S_GRADE(Sno, Avg_grade)。

```
INSERT INTO  S_GRADE (Sno, Avg_grade)
SELECT Sno, AVG(grade)
        FROM  SC
        WHERE Sno IN
        ( SELECT Sno
          FROM STUDENT
          WHERE sdept = 'IS')
        GROUP BY sno
        HAVING AVG(grade)>80;
```

3.5.1 插入数据

RDBMS在执行插入／更新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

3.5.2 删除数据

语法：DELETE FROM 表名 [WHERE 条件表达式]

作用：从表中删除符合WHERE子句中删除条件的元组；若WHERE子句缺省，则表示要删除表中的所有元组。

例4：删除学号为 ‘S001’的学生信息。

```
DELETE FROM STUDENT WHERE sno='S001';
```

例5：删除所有学生信息。

```
DELETE FROM STUDENT;
```

不带条件的DELETE语句与DROP TABLE语句的区别：

- DROP TABLE语句删除表和表中的所有数据；
- DELETE语句只删除表中的数据，表仍然保留。

3.5.2 删除数据

- 带子查询的删除语句

子查询同样也可以嵌套在DELETE语句中，用以构造执行删除操作的条件。

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);
```

3.5.3 更新数据

- 语法：

```
UPDATE 基本表名
SET 列名＝值表达式 [,列名＝值表达式…]
[WHERE 条件表达式]
```

- 作用：对表中满足WHERE条件的元组，按SET子句修改相应列的值。若WHERE子句缺省，则表示对所有元组进行修改。
- 例5：把所有学生的年龄加1。

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

- 例6：把课程号为‘C5’的课程名改为“电子商务”。

```
UPDATE COURSE
SET Cname='电子商务'
WHERE Cno = 'C5';
```

3.5.3 更新数据

- **带子查询的修改语句**

  子查询也可以嵌套在UPDATE语句中，用以构造执行修改操作的条件。

例　将计算机科学系全体学生的成绩加10分。

```
UPDATE SC
SET grade=grade+10
WHERE  'CS' =(SELECT Sdept
                FROM Student
                WHERE Student.Sno=SC.Sno);
```

(处理过程类似于相关子查询)

修改操作与数据库的一致性

UPDATE语句一次只能操作一个表。这会带来一些问题。

例如，学号为95007的学生因病休学一年，复学后需要将其学号改为96089，由于Student表和SC表都有关于95007的信息，因此两个表都需要修改，这种修改只能通过两条UPDATE语句进行。

- **第一条UPDATE语句修改Student表：**

```
UPDATE Student
    SET Sno='96089'
    WHERE Sno='95007';
```

- **第二条UPDATE语句修改SC表：**

```
UPDATE SC
    SET Sno='96089'
    WHERE Sno='95007';
```

必须保证这两条UPDATE语句要么都做，要么都不做。

事务

3.7 视图

视图是从一个或几个基本表（或视图）导出的一个虚表。

- 数据库中只存放视图的定义而不存放视图的数据，这些数据仍放在原来的基表中。当基表中的数据发生变化时从视图中查出的数据也随之改变了。
- 视图一经定义就可以对其进行查询，但对视图的更新操作有一定的限制。

3.7.1 视图的定义

1. 建立视图

语法：

```
CREATE  VIEW 视图名 [(列名[,列名]…)]
AS  子查询
[WITH  CHECK  OPTION]
```

3.7.1 视图的定义

说明：

- 视图的列名为可选项，缺省时，其列名即为子查询的查询目标列。但是在以下情况下，视图列名不可省略：
  - 视图由多个表连接得到，在不同的表中存在同名列，则需指定列名；
  - 当视图的列名为表达式或集函数的计算结果时，而不是单纯的属性名时，则需指明列名。
- 在子查询中不许使用ORDER BY子句，如果需要排序，则可在视图定义后，对视图查询时再进行排序。
- 如果指明了WITH CHECK OPTION选项，则在对视图进行更新时，所影响的元组必须满足视图定义中的WHERE条件。

3.7.1 视图的定义

- 行列子集视图：从一个基本表中导出，只是去掉了某些行或列(保留原表的主码)，这样的视图称为行列子集视图。
- 多表视图：从多个基本表或视图中导出。
- 带表达式的视图，即带虚拟列的视图。
- 分组视图，子查询带集函数和GROUP BY分组的视图。

3.7.1 视图的定义

例1：建立计算机学院（CS）学生视图，并要求进行修改和插入操作时仍需保证该视图只有CS系的学生（单表视图）。

```
CREATE VIEW CS_STUDENT AS
    SELECT sno, sname
    FROM  STUDENT
    WHERE sdept = 'CS'
    WITH CHECK OPTION;
```

例2：建立计算机学院选修5号课程的学生视图（多表视图）。

```
CREATE VIEW CS_S1(sno,sname,grade) AS
    SELECT STUDENT.sno,sname,grade
    FROM  STUDENT, SC
    WHERE sdept='CS'  AND
    STUDENT.sno=SC.sno  AND  cno='5';
```

3.7.1 视图的定义

例3：建立计算机学院选修了5号课程且成绩在90分以上的学生视图（视图之上建视图）。

```
CREATE VIEW CS_S2 AS
    SELECT sno, sname, grade
    FROM  CS_S1
    WHERE grade >= 90;
```

例4：建立学生出生年份的视图（表达式视图）。

```
create view student-year (sno, sname, sbirth)
AS
Select sno，  sname，   1999-sage
From student；
```

3.7.1 视图的定义

- 例5：求学生平均成绩视图（分组视图）。

```
CREATE VIEW Sc-AVG(SNO,FAVG)
AS
Select sno，AVG(grade)
from SC
Group By Sno;
```

- 例6:将Student表中所有女生记录定义为一个视图（不指定属性列）

```
CREATE VIEW F_Stu(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_Stu视图的映象关系被破坏，导致该视图不能正确工作。

3.7.1 视图的定义

2. 删除视图

- 语法：

```
DROP  VIEW <视图名> ［CASCADE］;
```

- 作用：从数据库中删除一个视图的定义信息。
- CASCADE表示把该视图和它导出的视图一起删除。
- 例如：撤消视图CS_S2。

```
DROP VIEW CS_S2;
```

- 注：视图删除后，只会删除该视图在数据字典中的定义，而与该视图有关的基本表中的数据不会受任何影响。

3.7.2 查询视图

- 用户角度：查询视图与查询基本表相同
- DBMS执行对视图的查询时，
  - 首先进行有效性检查，检查查询涉及的表、视图等是否在数据库中存在；
  - 如果存在，则从数据字典中取出查询涉及的视图的定义；
  - 把定义中的子查询和用户对视图的查询结合起来，转换成等价的基本表的查询；
  - 然后再执行这个经过修正的查询。

将对视图的查询转换为对基本表的查询的过程称为视图的消解（View Resolution）。

3.7.2 查询视图

例：查询计算机学院学习了5号课程且成绩为95分的学生学号和姓名。

方案1：从基本表中查询。

```
SELECT STUDENT.sno,sname
FROM  STUDENT, SC
WHERE STUDENT.sno=SC.sno AND sdept='CS'
AND cno='5' AND grade=95 ;
```

方案2：利用视图CS_S1查询。

```
SELECT  sno, sname
FROM CS_S1
WHERE  grade=95;
```

由此可见，利用视图可以简化复杂的查询语句。

3.7.2 查询视图

- 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。

[例]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,
 AS
SELECT  Sno，AVG(Grade)
FROM  SC
GROUP BY Sno;
```

查询执行转换：

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;
```

3.7.3 更新视图

- 视图的更新操作包括插入、修改和删除数据，其语法格式如同对基本表的更新操作一样。
- 由于视图是一张虚表，所以对视图的更新，最终实际上是转换成对基本表的更新。

- ［例］对视图CS_STUDENT的更新：

```
INSTER  INTO CS_STUDENT VALUES (‘04008’ , ‘张立’ );
```

在执行时将转换为对表STUDENT的更新：

```
INSERT INTO STUDENT
 VALUES (‘04008’, ‘张立’ , NULL, NULL, NULL);
```

3.7.3 更新视图

[例] 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' )；
```

3.7.3 更新视图

- 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。

3.7.3 更新视图

视图更新的约束：

根据视图的定义可将视图分为**可更新视图**和**不允许更新的视图**。如下列情况的视图为不允许更新的视图：

- 字段由表达式或常数组成，不允许执行INSERT和该字段上的UPDATE，但可以执行DELETE；
- 字段由集函数组成，不允许更新；
- 视图定义含有GROUP BY或DISTINCT，不允许更新；
- 有嵌套查询，且内外查询使用相同表时，不允许更新；
- 定义中有多表联接时，不允许更新；
- 由不允许更新的视图导出的视图，不允许更新。

3.7.3 更新视图

视图的作用

- 简化用户的操作
- 使用户能以多种角度看待同一数据
- 对重构数据库提供一定程度的逻辑独立性
- 是数据共享与保密的一种安全机制
- 适当的利用视图可以更清晰的表达查询

小结

- 本章介绍SQL语言的使用方法。
- 在讲解SQL语言的同时，进一步介绍了关系数据库的有关概念，如索引和视图的概念及其作用。
- SQL语言具有数据定义、数据查询、数据更新、数据控制四大功能。其中，数据查询功能最为丰富和复杂，也非常重要，初学者掌握起来有一定的困难，应反复上机加强练习。
  - SQL Server 2008+
  - DM数据库： www.dameng.com DM6+
  - MySQL

本章作业：P130 3，4，5