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）

福州大学软件学院

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

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）

- INTO子句
  - 属性列的顺序可与表定义中的顺序不一致
  - 没有指定属性列
  - 指定部分属性列
- VALUES子句
  - 提供的值必须与INTO子句匹配
    - 值的个数
    - 值的类型

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）

- 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
  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；
```

2. 修改多个元组的值

[例6_a] 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'=
        (SELECT   Sdept
         FROM     Student
         WHERE    Student.Sno = SC.Sno)；
```

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。

解法2

```
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;
```

2. 删除多个元组的值

[例9_a] 删除2号课程的所有选课记录。

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

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)；
```

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。

解法2

```
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

3.6 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

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’；
```

行列子集视图视图IS_Student由Sno，Sname，Sage三列组成

查询视图（续）

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）

[例2_a] 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;
```

建立视图（续）

DY ： Day of week abbreviated Mon, Tue, Fri
DAY ： Day of week spelled out Monday, Tuesday, Friday
D ： Day of week (1–7) 1,2,3,4,5,6,7——注意：每星期的第1天是”星期日“
DD ： Day of month (1–31) 1,2,3,4…31
DDD ： Day of year (1–366) 1,2,3,4…366

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

  - 在设计数据库时，为了减少数据冗余，基本表中只存放基本数据，由基本数据经过各种计算派生出的数据一般是不存储的。

  - 视图中的数据并不实际存储，所以定义视图时可以根据应用的需要，设置一些派生属性列，以方便应用程序的编制。

建立视图（续）

- 带表达式的视图
  - 派生属性称为虚拟列。带虚拟列的视图称为带表达式的视图。
  - 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，2004-Sage
FROM    Student；
```

建立视图（续）

- 分组视图

  - 用带集函数和GROUP BY子句的查询来定义的视图称为分组视图

  - 分组视图必须明确定义组成视图的各个属性列名

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

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

更新视图（续）

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）

❖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。

建立视图（续）

[例7_a]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student2(F_Sno，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。

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
  - 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

由于IS_S1视图导出了IS_S2视图

➢级联删除：

DROP VIEW IS_S1 CASCADE;

/* 删除了视图IS_S1和由它导出的所有视图*/

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

3.6.2 查询视图

- 从用户角度而言，查询视图与查询基本表的方法相同
- DBMS实现视图查询的方法
  - 视图实体化法（View Materialization）
    - 进行有效性检查，检查所查询的视图是否存在。如果存在，则从数据字典中取出视图的定义
    - 执行视图定义，将视图临时实体化，生成临时表
    - 将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) new

3.6.2 查询视图

- 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

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；
```

IS_Student视图的定义 (参见视图定义例1)

建立视图（续）

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）

- 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；
```

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）

[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

建立视图（续）

更新视图（续）

- 视图的可更新性
  - 行列子集视图是可更新的。
  - 除行列子集视图外，还有些视图理论上是可更新的，但它们的确切特征还是尚待研究的课题。
  - 还有些视图从理论上是不可更新的。

更新视图（续）

- 不可更新的视图与不允许更新的视图是两个不同的概念
- 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。

更新视图（续）

(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。

(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。

更新视图（续）

(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。

例：视图GOOD_SC(修课成绩在平均成绩之上的元组)

```
CREATE VIEW GOOD_SC
    AS
      SELECT  Sno，Cno，Grade
      FROM SC
      WHERE Grade >
        (SELECT AVG(Grade)
         FROM    SC)；
```

更新视图（续）

(7) 一个不允许更新的视图上定义的视图也不允许更新。

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

3.6.4 视图的作用

- 视图最终是定义在基本表之上的，对视图的一切操作最终也要转换为对基本表的操作。而且对于非行列子集视图进行查询或更新时还有可能出现问题。

3.6.4 视图的作用

- 合理使用视图能够带来许多好处
  - 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

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

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

将学生关系

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

下课了。。。

休息一会儿。。。