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

数据更新

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
  - 插入元组
  - 插入子查询结果
    - 可以一次插入多个元组

1. 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
  - 指定要插入数据的表名及属性列
  - 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：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
  - 指定部分属性列：插入的元组在其余没有指定的属性列上取空值或者默认缺省值

插入元组（续）

- 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3.69]将一个新学生元组（学号：201215128;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 ('201215128','陈冬','男','IS',18);
```

插入元组（续）

[例3.70]将学生张成民的信息插入到Student表中。

```
INSERT INTO  Student
VALUES ('201215126','张成民','男’,18,'CS');
```

插入元组（续）

[例3.71] 插入一条选课记录（ '201215128','1 '）。

```
INSERT  INTO SC(Sno,Cno)
   VALUES ('201215128 ',' 1 ');
```

关系数据库管理系统将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT  INTO SC
   VALUES (' 201215128 ',' 1 ',NULL);
```

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;

- **INTO**子句
- 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3.72] 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*/
```

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）

- 关系数据库管理系统在执行插入（修改、删除）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

3.5.2 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
- SET子句给出<表达式>的值用于取代相应的属性列
- 如果省略WHERE子句，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3.73] 将学生201215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 201215121 ';
```

2. 修改多个元组的值

[例3.74] 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3.75] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC
SET    Grade=0
WHERE Sno  IN
      (SELETE Sno
       FROM    Student
       WHERE  Sdept= 'CS' );
```

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
  - 无该子句将会删除表中的全部元组

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3.76] 删除学号为201215128的学生记录。

```
DELETE  FROM Student
WHERE Sno= '201215128';
```

2. 删除多个元组的值

[例3.77] 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3.78] 删除计算机科学系所有学生的选课记录。

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

空值处理

3.6 空值的处理

- 空值就是“不知道”或“不存在”或“无意义”的值。
- 一般有以下几种情况：
  - 该属性应该有一个值，但目前不知道它的具体值
  - 该属性不应该有值
  - 由于某种原因不便于填写

1. 空值的产生

- 空值是一个很特殊的值，含有不确定性。对关系运算带来特殊的问题，需要做特殊的处理。
- 空值的产生有其实际需求

  学生在选课后，产生选课表，但是还没有成绩。这时候成绩部分就为空值，它和0不一样（不是0分）

空值的产生（续）

[例 3.79]向SC表中插入一个元组，学生号是”201215126”，课程号是”1”，成绩为空。

```
INSERT INTO SC(Sno,Cno,Grade)
VALUES('201215126 ','1',NULL);   /*该学生还没有考试成绩，取空值*/
```

或

```
INSERT INTO SC(Sno,Cno)
VALUES(' 201215126 ','1');          /*没有赋值的属性，其值为空值*/
```

空值的产生（续）

[例3.80] 将Student表中学生号为”201215200”的学生所属的系改为空值。

```
UPDATE Student
SET Sdept = NULL
WHERE Sno='201215200';
```

2. 空值的判断

❖判断一个属性的值是否为空值，用IS NULL或IS NOT NULL来表示。

[例 3.81] 找出漏填了性别或者年龄信息的记录

```
SELECT *
FROM Student
WHERE Ssex IS NULL OR Sage IS NULL ;
```

3. 空值的约束条件

- 属性定义（或者域定义）中
  - 有NOT NULL约束条件的不能取空值
  - 加了UNIQUE限制的属性不能取空值
  - 码属性不能取空值

4. 空值的算术运算、比较运算和逻辑运算

- 空值与另一个值（包括另一个空值）的算术运算的结果为空值
- 空值与另一个值（包括另一个空值）的比较运算的结果为**UNKNOWN**。
- 有**UNKNOWN**后，传统二值（**TRUE**，**FALSE**）逻辑就扩展成了三值逻辑

空值的算术运算、比较运算和逻辑运算(续)

表3.8　逻辑运算符真值表

x　y	x AND y	x OR y	NOT x
T　T	T	T	F
T　U	U	T	F
T　F	F	T	F
U　T	U	T	U
U　U	U	U	U
U　F	F	U	U
F　T	F	T	T
F　U	F	U	T
F　F	F	F	T

T表示TRUE，F表示FALSE，U表示UNKNOWN

空值的算术运算、比较运算和逻辑运算（续）

[例3.82] 找出选修1号课程的不及格的学生。

```
SELECT Sno
FROM SC
WHERE Grade < 60 AND Cno='1';
```

查询结果不包括缺考的学生，因为他们的Grade值为null。

空值的算术运算、比较运算和逻辑运算（续）

[例 3.83] 选出选修1号课程的不及格的学生以及缺考的学生。

```
SELECT Sno
FROM SC
WHERE Cno='1' AND (Grade<60 OR Grade IS NULL);
```

聚集函数：MAX、MIN、AVG、COUNT、SUM
遇到空值时，除了COUNT(*)外，都跳过空值而去处理非空值。

3.7 视图（外模式）

❖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

3.7 视图

3.7.1 定义视图

3.7.2 查询视图

3.7.3 更新视图

3.7.4 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];
```

建立视图（续）

- **WITH CHECK OPTION**
  - 对视图进行UPDATE，INSERT和DELETE操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）注：一般只允许对行列子集视图进行更新
- 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某些列，但保留了主码，我们称这类视图为行列子集视图。

建立视图（续）

- 组成视图的属性列名：全部省略或全部指定
  - 全部省略:
    - 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:
    - 某个目标列是聚集函数或列表达式
    - 多表连接时选出了几个同名列作为视图的字段
    - 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）

- 关系数据库管理系统执行**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，首先判断视图是否存在，如果存在，则从数据字典中取出视图的定义，把定义中的子查询和对视图的查询结合起来，转换成等价的对基本表的查询。

对象 VIEWS @information_...

开始事务 备注 筛选 排序 导入 导出

TABLE_CATALO	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	C
def	choose	choose_1_view	select `choose`.`choose`.`choose_no` AS `choose_no`,`choose`.`choose`.`student_no` AS	N
def	choose	choose_2_view	select `choose`.`choose`.`choose_no` AS `choose_no`,`choose`.`choose`.`student_no` AS	L
def	choose	student_view	select `choose`.`student`.`student_name` AS `student_name`,`choose`.`classes`.`class_nan	N

建立视图（续）

[例3.84] 建立信息系学生的视图。

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

建立视图（续）

[例3.85]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满足信息系学生这个条件。

```
CREATE VIEW IS_Student1
AS
SELECT Sno,Sname,Sage
FROM  Student
WHERE  Sdept= 'IS'
WITH CHECK OPTION;
```

IS_Student1视图是一个行列子集视图。

建立视图（续）

❖基于多个表的视图

[例3.86] 建立信息系统选修了1号课程的学生的视图（包括学号、姓名、成绩）。

```
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3.87] 建立信息系统选修了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

[例3.88] 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,2014-Sage
FROM  Student;
```

建立视图（续）

❖分组视图

[例3.89] 将学生的学号及平均成绩定义为一个视图

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

2. 删除视图

❖语句的格式：

DROP VIEW <视图名>[CASCADE];

- 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3.91] 删除视图BT_S和IS_S1

```
DROP VIEW BT_S;     /*成功执行*/
DROP VIEW IS_S1;    /*拒绝执行*/
```

要删除IS_S1，需使用级联删除：

```
DROP VIEW IS_S1 CASCADE;
```

3.7 视图

3.7.1 定义视图

3.7.2 查询视图

3.7.3 更新视图

3.7.4 视图的作用

3.7.2 查询视图

- 用户角度：查询视图与查询基本表相同
- 关系数据库管理系统实现视图查询的方法
  - 视图消解法（View Resolution）
    - 进行有效性检查
    - 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3.92] 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;
```

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3.93] 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的查询。？此说法待商榷

[例3.94]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视图的定义如下：

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

查询视图（续）

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;
```

查询视图（续）

[例3.94]也可以用如下SQL语句完成

```
SELECT *
FROM  (SELECT Sno,AVG(Grade) Gavg
          FROM  SC
         GROUP BY Sno) AS S_G    WHERE Gavg>=90;
```

3.7 视图

3.7.1 定义视图

3.7.2 查询视图

3.7.3 更新视图

3.7.4 视图的作用

更新视图（续）

[例3.95] 将信息系学生视图IS_Student中学号”201215125”的学生姓名改为”刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= ' 201215125 ';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= ' 201215122 ' AND Sdept= 'IS';
```

更新视图（续）

[例3.96] 向信息系学生视图IS_Student中插入一个新的学生记录，其中学号为”201215129”，姓名为”赵新”，年龄为20岁

```
INSERT
INTO IS_Student
VALUES(‘201215129’,’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' );
```

注意：不同的系统处理方式不一样，MySQL会在系名处自动填入NULL或者默认值

更新视图（续）

❖WITH CHECK OPTION的限定

视图定义：CREATE VIEW IS_Student1 AS SELECT Sno,Sname,Sage
FROM Student WHERE Sdept= 'IS‘ WITH CHECK OPTION;

INSERT
INTO IS_Student1
VALUES(‘201215130’,’赵新’,20);

```
mysql> INSERT
    -> INTO IS_Student1
    -> VALUES('201215130','赵新',20);
ERROR 1369 (HY000): CHECK OPTION failed 's_t.is_student1'
```

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：例3.89定义的视图S_G为不可更新视图。

```
UPDATE  S_G
SET     Gavg=90
WHERE  Sno= ‘201215121’;
```

  这个对视图的更新无法转换成对基本表SC的更新

更新视图（续）

- 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

更新视图（续）

- **DB2对视图更新的限制：**
  - 若视图是由两个以上基本表导出的，则此视图不允许更新。
  - 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。
  - 若视图的字段来自 聚集函数，则此视图不允许更新。
  - 若视图定义中含有**GROUP BY**子句，则此视图不允许更新。
  - 若视图定义中含有**DISTINCT**短语，则此视图不允许更新。
  - 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。

3.7 视图

3.7.1 定义视图

3.7.2 查询视图

3.7.3 更新视图

3.7.4 视图的作用

3.7.4 视图的作用

- 视图能够简化用户的操作
- 视图使用户能以多种角度看待同一数据
- 视图对重构数据库提供了一定程度的逻辑独立性
- 视图能够对机密数据提供安全保护
- 适当的利用视图可以更清晰的表达查询

视图的作用（续）

- 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

视图的作用（续）

- 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

视图的作用（续）

❖视图对重构数据库提供了一定程度的逻辑独立性

■数据库重构：

例：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)

视图的作用（续）

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，用户的应用程序通过视图仍然能够查找数据

视图的作用（续）

- 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

视图的作用（续）

- 适当的利用视图可以更清晰的表达查询
  - 经常需要执行这样的查询“对每个同学找出他获得最高成绩的课程号”。可以先定义一个视图，求出每个同学获得的最高成绩

```
CREATE VIEW VMGRADE
 AS
    SELECT Sno, MAX(Grade)  Mgrade
    FROM  SC
    GROUP BY Sno;
```

视图的作用（续）

然后用如下的查询语句完成查询：

```
SELECT SC.Sno,Cno
FROM SC,VMGRADE
WHERE SC.Sno=VMGRADE.Sno AND
SC.Grade=VMGRADE .Mgrade;
```