第3章 关系数据库标准语言SQL

（§3.5 -- §3.6）

哈尔滨工业大学（威海）

本章内容提要

- ◆3.1 SQL概述
- ◆3.2 学生-课程数据库
- ◆3.3 数据定义
- ◆3.4 数据查询
- ◆*3.5 数据更新*
- ◆3.6 视图

3.5 数据更新

- 插入数据
- 修改数据
- 删除数据

3.5 数据更新

- ***插入数据***
- 修改数据
- 删除数据

3.5 数据更新-插入数据

◆插入数据的方式

- ➢插入元组
- ➢插入子查询结果
  - ✓可以一次插入多个元组

3.5 数据更新-插入数据

◆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

3.5 数据更新-插入数据

◆插入元组

- 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

3.5 数据更新-插入数据

◆插入元组

- ［例1］ 将一个新学生元组（学号：200215128；姓名：陈冬；性别：男；所在系：CS；年龄：18岁）插入到Student表中。

```
INSERT
INTO Student (Sno，Sname，Ssex，Dno，Sage)
VALUES ('200215128'，'陈冬'，'男'，'CS'，18);
```

3.5 数据更新-插入数据

◆插入元组

- ［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');
```

3.5 数据更新-插入数据

◆插入元组

➢ ［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);
```

3.5 数据更新-插入数据

◆插入子结果查询

- 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

3.5 数据更新-插入数据

◆插入子结果查询

- 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

3.5 数据更新-插入数据

◆插入子结果查询

- [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
    (Dno CHAR(15),          /* 系号*/
     Avg_age SMALLINT);     /*学生平均年龄*/
```

3.5 数据更新-插入数据

◆插入子结果查询

- [例4] 对每一个系，求学生的平均年龄，并把结果存入数据库。

第二步：插入数据

```
INSERT
 INTO  Dept_age(Dno，Avg_age)
    SELECT  Dno，AVG(Sage)
    FROM  Student
    GROUP BY Dno;
```

3.5 数据更新-插入数据

◆插入子结果查询

- [例5]将平均成绩大于90的学生加入到EXCELLENT中。EXCELLENT表结构为EXCELLENT(Sno,avgScore)，其中avgScore为平均成绩

```
insert  into  EXCELLENT ( Sno, avgScore)
        select   Sno , avg(grade)
        from    SC
        group  by (Sno)
        having  avg(grade) > 90;
```

3.5 数据更新-插入数据

◆插入子结果查询

- ➢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

- 插入数据
- *修改数据*
- 删除数据

3.5 数据更新-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子句条件的元组的指定列值

3.5 数据更新-修改数据

- **SET子句**
  - 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

3.5 数据更新-修改数据

◆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

3.5 数据更新-修改数据

◆修改某一元组的值

[例6] 将学生200215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 200215121 ';
```

3.5 数据更新-修改数据

◆修改多个元组的值

[例7] 将所有教师工资上调5%

```
Update  Prof
Set   Sal  = Sal * 1.05  ;
```

3.5 数据更新-修改数据

◆修改多个元组的值

[例8] 将所有学生的年龄增加1岁

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

3.5 数据更新-修改数据

◆带子查询的修改语句

[例9] 将MA系全体学生的成绩置零。

```
UPDATE SC
 SET  Grade=0
 WHERE ‘MA'=
          (SELECT Dno
           FROM  Student
           WHERE  Student.Sno = SC.Sno)；
```

3.5 数据更新-修改数据

◆带子查询的修改语句

[例10]将所有计算机系的教师工资上调10%。

```
Update  Prof
 Set   Sal  = Sal * 1.1
 Where  Dno  in
     ( Select   Dno
       From  Dept
       Where      Dname = '计算机');
```

3.5 数据更新-修改数据

◆带子查询的修改语句

[例11]当某同学1号课的成绩低于该课程平均成绩时，将该同学该门课成绩提高5%。

```
Update  SC
 Set   Grade  = Grade * 1.05
 Where   Cno= '1'  and  Grade<
          ( Select   AVG(Grade)
            From SC
           Where Cno = '1' ) ;
```

3.5 数据更新-修改数据

◆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

3.5 数据更新

- 插入数据
- 修改数据
- ***删除数据***

3.5 数据更新-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

3.5 数据更新-删除数据

◆语句格式

DELETE

FROM　<表名>

[WHERE <条件>]；

◆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

3.5 数据更新-删除数据

◆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

3.5 数据更新-删除数据

◆删除某一元组的值

[例13] 删除学号为200215128的学生记录。

```
DELETE
FROM Student
WHERE Sno= '200215128 ' ;
```

3.5 数据更新-删除数据

◆删除多个元组的值

[例14]删除98030101号同学所选的所有课程。

```
DELETE
FROM SC
WHERE Sno= '98030101' ;
```

3.5 数据更新-删除数据

◆删除多个元组的值

***[例15] 删除所有的学生选课记录。

```
DELETE
 FROM SC；
```

3.5 数据更新-删除数据

◆带子查询的删除语句

[例16] 删除CS系所有学生的选课记录。

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

3.5 数据更新-删除数据

◆带子查询的删除语句

[例17] 删除有四门不及格课程的所有同学信息。

```
Delete
From  Student
Where  Sno  in
 ( Select   Sno  From  SC
  Where  Grade < 60
   Group by   Sno
   Having    Count(*)>= 4 );
```

3.5 数据更新-删除数据

◆带子查询的删除语句

[例18] 删除低于平均工资的老师记录。

```
delete
from    Prof
where    Sal  <
              (select   avg(Sal)
               from     Prof)；
```

本章内容提要

- ◆3.1 SQL概述
- ◆3.2 学生-课程数据库
- ◆3.3 数据定义
- ◆3.4 数据查询
- ◆3.5 数据更新
- ◆*3.6 视图*

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

- 定义视图
- 查询视图
- 更新视图
- 视图的作用

3.6 视图

- ***定义视图***
- **查询视图**
- **更新视图**
- **视图的作用**

3.6 视图-定义视图

◆建立视图

➢语句格式

CREATE VIEW

<视图名> [(<列名> [，<列名>]…)]

AS <子查询>

[*WITH CHECK OPTION*];

3.6 视图-定义视图

◆建立视图

- 组成视图的属性列名：全部省略或全部指定（没有第三种选择）

如果省略了视图的各个属性名，则隐含该视图由子查询中SELECT子句目标列中的诸字段组成

但下面三种情况下必须明确指定组成视图的所有列名：

（1）某个目标列不是单纯的属性名，而是聚集函数或列表达式；

（2）多表连接时选出了几个同名列作为视图的字段；

（3）需要在视图中为某个列启用新的更合适的名字。

- 子查询不允许含有**ORDER BY**子句和**DISTINCT**短语
- *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。

3.6 视图-定义视图

◆建立视图

- [例1] 建立IS系学生的视图。

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

3.6 视图-定义视图

◆建立视图

- [例2]建立IS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IS系的学生

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

3.6 视图-定义视图

◆建立视图

➢带WITH CHECK OPTION选项时对视图的更新操作

✓插入操作，插入的记录在刷新视图后必须可以看到；

✓修改操作，修改完的结果也必须能通过该视图看到；

✓如果删除，当然只能删除视图里有显示的记录。

3.6 视图-定义视图

◆建立视图

- 例如：

  insert into IS_Student values('6','晓东',24)

  出错！

- 由于行通过视图进行添加或更新，当其不再符合定义视图的查询的条件时，它们即从视图范围中消失。

3.6 视图-定义视图

◆建立视图

➢又如：

```
CREATE VIEW IS_StudentVIEW
  AS
 SELECT Sno，Sname，Sage，Dno
 FROM  Student
WHERE Dno = ‘IS'
  WITH CHECK OPTION;

        update IS_StudentVIEW
        set Dno=‘CS'
        where sno=‘2002101‘
```

出错！

3.6 视图-定义视图

◆建立视图

➢[例3] 建立计算机系学生的视图。

```
CREATE VIEW CompStud
    AS
    SELECT Sno，Sname，Ssex，Sage，Dno
    FROM  Student
    WHERE  Dno in
                ( Select   Dno  From  Dept
                  Where  Dname = '计算机' ) ;
```

3.6 视图-定义视图

◆建立视图

➢基于多个基表的视图

✓[例4] 建立IS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  Dno= ‘IS' AND
          Student.Sno=SC.Sno AND
          SC.Cno= '1';
```

3.6 视图-定义视图

◆建立视图

- ➢基于视图的视图
  - ✓[例5] 建立IS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；
```

3.6 视图-定义视图

◆建立视图

➢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，2014-Sage
FROM  Student;
```

3.6 视图-定义视图

◆建立视图

➢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。

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

3.6 视图-定义视图

◆建立视图

➢不指定属性列

✓[例8]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student(Sno，
                      name，sex，age，Dno)
 AS
SELECT *
 FROM  Student
WHERE Ssex=‘女’;
```

3.6 视图-定义视图

◆建立视图

➢不指定属性列

✓[例8]将Student表中所有女生记录定义为一个视图

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。

3.6 视图-定义视图

◆删除视图

➢语句格式

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

3.6 视图-定义视图

◆删除视图

➢［例9］ 删除视图BT_S：

```
DROP VIEW BT_S;
```

删除视图IS_S1：

```
DROP VIEW IS_S1;
```

✓拒绝执行

✓级联删除：

```
DROP VIEW IS_S1 CASCADE;
```

3.6 视图

- 定义视图
- ***查询视图***
- 更新视图
- 视图的作用

3.6 视图-查询视图

◆查询视图

- ➢用户角度：查询视图与查询基本表相同
- ➢RDBMS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

3.6 视图-查询视图

[例10] 在IS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  Dno= ‘IS'  AND  Sage<20；
```

3.6 视图-查询视图

[例11] 查询选修了1号课程的IS系学生

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

3.6 视图-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。

3.6 视图-查询视图

[例12]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：

```
SELECT  Sno，AVG(Grade)
FROM  SC
GROUP BY Sno;
```

3.6 视图-查询视图

◆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-查询视图

- 目前多数关系数据库系统对行列子集视图的查询均能正确转换。但对非行列子集视图的查询就不一定能做转换了，因此这类查询应该直接对基本表进行。
- ***行列子集视图***：若视图是从单个基本表导出，并且只是去掉了基本表的某些行和某些列，但保留了主码。

3.6 视图

- 定义视图
- 查询视图
- ***更新视图***
- 视图的作用

3.6 视图-更新视图

[例13] 将IS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 Dno= ‘IS';
```

3.6 视图-更新视图

[例14] 向IS系学生视图IS_Student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)
VALUES (‘200215129 '，'赵新'，20)；
```

3.6 视图-更新视图

[例15]删除IS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 Dno= ‘IS';
```

3.6 视图-更新视图

- 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
  - 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

3.6 视图

- 定义视图
- 查询视图
- 更新视图
- ***视图的作用***

3.6 视图-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

本章小结

- SQL是关系数据库语言的工业标准。各个数据库厂商支持的SQL语言在遵循标准的基础上常常做不同的扩充或修改。
- SQL语言既可以作为交互式数据库语言使用，也可以作为程序设计语言的子语言使用。
- SQL语言可以分为***数据定义***、***数据查询***、***数据更新***、数据控制四大部分。
- 视图是关系数据库系统中的重要概念，合理使用视图具有很多优点。