



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



内蒙古农业大学计算机与信息工程学院



3.6 视图

- ◆ 学习目标
 - 理解并描述什么是视图
 - 掌握并能够熟练使用SQL视图创建语句创建视图
 - 掌握并能够熟练使用SQL语句进行基于视图的查询和更新
 - 理解并描述视图的作用



3.6 视图

- ◆ 出于安全考虑，让所有用户都看到全局逻辑模型是不合适的，可能需要向用户隐藏特定的数据
- ◆ 在实际的数据库系统中，用户只关心或只被允许使用数据库中某些基本关系表或基本关系表中的某些属性
- ◆ 这些数据就构成了数据库的外模式，外模式的定义可以通过定义视图来实现



3.6 视图

视图

- 是从一个或几个基本表（或视图）导出的**虚表**
- 数据库中只存放视图的**定义**，视图对应的数据仍存放在原来的基本表中
- 基表中的数据发生变化后，从视图中查询出的数据也随之改变
- 视图为用户提供了一个观察底层数据的窗口，透过它用户可以看到数据库中自己感兴趣的数据及其变化



1. 定义视图

- 1. 建立视图语句格式

CREATE VIEW <视图名>

[(<属性列名1> [, <属性列名2>]...)]

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



1. 定义视图

- 组成视图的属性列名：**全部省略或全部指定**
 - ◆ 全部省略：
 - ◆ 子查询中**SELECT**目标列只是单纯的属性名，组成视图的属性列名由**SELECT**目标列中的诸字段组成
 - ◆ 明确指定视图的所有列名：
 - ◆ 子查询**SELECT**目标列中含有聚集函数或列表达式
 - ◆ 子查询**SELECT**目标列中含对多表查询产生的同名属性列
 - ◆ 需要在视图中为某个列启用新的更合适的名字



1. 定义视图

- AS子查询可以是任意复杂的**SELECT**语句，是否可以含有**GROUP BY**子句和**DISTINCT**短语，取决于具体的**DBMS**是否支持
- **WITH CHECK OPTION**子句
 - 要求透过视图进行增删改操作时，要保证更新、插入或删除的行满足视图定义子查询中的谓词条件（即子查询中的元组选择条件）



1. 定义视图

- 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**子查询**语句
- 在对视图查询时，才按视图的定义从基本表中将数据查出



1. 定义视图

[例1] 建立20160101班学生的视图（属性列包括学号、姓名、性别和出生年月）

```
CREATE VIEW Stu_160101
AS
SELECT Sno, Sname, ssex, birthdate
FROM Student
WHERE clno='20160101'
```

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，这类视图称为**行列子集视图**



1. 定义视图

[例2] 建立20160101班学生的视图，并要求透过该视图进行的更新操作只涉及20160101班学生。

```
CREATE VIEW Stu_160101_1  
AS  
SELECT Sno, Sname, ssex, birthdate  
FROM Student  
WHERE cjno='20160101'  
with check option
```

with check option的视图

透过该视图进行增删改操作时，RDBMS会自动加上cjno='20160101'



1. 定义视图

[例3] 建立20160101班选修了001号课程的学生视图（属性列包括学号，姓名和成绩）

```
CREATE VIEW S1_160101
AS
SELECT Student.Sno, Sname, Grade
FROM Student, SC
WHERE c1no='20160101' AND Student.Sno=SC.Sno
AND SC.Cno='001'
```

基于多个基本表的视图



1. 定义视图

[例4] 建立20160101班选修了001号课程且成绩在90分以上的学生的视图。

```
CREATE VIEW S2_160101  
AS  
SELECT Sno, Sname, Grade  
FROM S1_160101  
WHERE Grade>=90
```

建在一个或多个已定义好的视图上的视图称为基于视图的视图



1. 定义视图

[例5] 定义一个反映学生年龄的视图（属性包括学号、姓名和年龄）

```
CREATE VIEW Age_Stu(Sno, Sname, age)
AS
SELECT Sno, Sname, year(getdate())-year(birthdate)
FROM Student
```

- ◆ 设置一些派生属性列, 也称为虚拟列--age
- ◆ 带虚拟列的视图也称为带表达式的视图
- ◆ 带表达式或聚集函数的视图或者明确定义组成视图的各个属性列名, 或者在select子句中指定列别名



1. 定义视图

[例6]将学生的学号及其选修课程的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
```

用带有聚集函数和**GROUP BY** 子句的查询来定义的视图称为**分组视图**。



1. 定义视图

[例7]将Student表中所有女生记录定义为一个视图。

定义语句1:

```
CREATE VIEW  
F_Student1 (stdnum, name, sex, birthdate, adress, clno)  
AS  
SELECT *  
FROM Student  
WHERE Ssex='女'
```

以 **SELECT *** 方式创建的视图**可扩充性**差，应尽可能避免因为修改基表Student的结构后，Student表与F_Student1视图的映射关系被破坏，导致该视图不能正确工作。



1. 定义视图

定义语句2:

```
CREATE VIEW F_Student2  
AS  
SELECT  
sno, sname, ssex, birthdate, clno, adress  
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。



2. 删除视图

- **DROP VIEW** <视图名>[(cascade)];
 - 视图删除后视图的定义将从数据字典中删除
 - 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均无法使用，但视图的定义并没有从数据字典中清除。



3. 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)



3. 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



3. 查询视图

[例1] 在20160101班学生的视图中找出年龄小于20岁学生。

```
SELECT *  
FROM Stu_160101  
WHERE year(getdate())-year(birthdate)<20
```

```
CREATE VIEW Stu_160101  
AS  
SELECT Sno, Sname, ssex, birthdate  
FROM Student  
WHERE clno='20160101'
```



3. 查询视图

视图消解法转换后的查询语句为：

```
SELECT *  
FROM Student  
WHERE c1no='20160101' AND  
       year(getdate())-year(birthdate)<20
```



3. 查询视图

[例2] 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
```


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




3. 查询视图

查询转换:

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade) >=90
```



```
SELECT Sno, AVG(Grade)
FROM SC
WHERE AVG(Grade) >=90
GROUP BY Sno
```



WHERE子句中不能用聚集函数作为条件表达式。



3. 查询视图

第二种方法:

```
SELECT *  
FROM (select sno, avg(grade) from sc group by sno)  
      as avg_sc(sno, gavg)  
Where gavg >= 90
```

思考：定义视图并查询视图和基于派生表的查询的区别？

- ◆ 视图定义后将永久保存在数据字典中，所有查询直接引用该视图
- ◆ 派生表只是在语句执行时临时定义，语句执行后该定义即被删除



4. 更新视图

- 用户角度：更新视图与更新基本表相同
- 因为视图是不实际存储数据的虚表，因此对视图的更新最终要转换为对基本表的更新。
- 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



4. 更新视图

[例1] 将20160101班学生视图Stu_160101中学号2016010101的学生姓名改为“刘辰”。

```
UPDATE Stu_160101  
SET Sname='刘辰'  
WHERE Sno='2016010101'
```

转换后的语句:

```
UPDATE Student  
SET Sname='刘辰'  
WHERE Sno='2016010101'  
AND clno='20160101'
```

```
CREATE VIEW Stu_160101  
AS  
SELECT Sno, Sname, ssex, birthdate  
FROM Student  
WHERE clno='20160101'
```



4. 更新视图

[例2] 向20160101班学生视图中插入一个新的学生记录：
2016010103, 赵新, 男, 1994-05-12

```
INSERT  
INTO Stu_160101  
VALUES ('2016010103', '赵新', '男', '1994-05-12')
```

- 如果查看视图，则无法在视图上查到该学生的详细信息
- DBMS并没有将对20160101班学生视图的插入转换为向基本表插入一个20160101班的学生操作
- 在定义视图时可以使用with check option避免该情况的发生



4. 更新视图

[例3] 删除视图Stu_160101中学号为2016010103的记录

```
DELETE  
FROM Stu_160101  
WHERE Sno='2016010103'
```

转换为对基本表的更新:

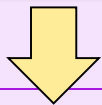
```
DELETE  
FROM Student  
WHERE Sno='2016010103' and c1no='20160101'
```



4. 更新视图

- 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

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



```
UPDATE S_G
SET Gavg=90
WHERE Sno='2016010101'
```

该更新语句对视图的更新无法转换成对基本表SC的更新



实际系统对视图更新的限制

- 允许对行例子集视图进行更新
- 对其他类型视图的更新不同系统有不同限制

SQL SERVER对视图更新的限制:

- (1) 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。
- (2) 由多表导出的视图不允许更新



实际系统对视图更新的限制

- (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) 一个不允许更新的视图上定义的视图也不允许更新。
- (7) 基本表中没有在视图中出现的属性无NOT NULL约束时，才允许进行插入操作。



5. 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一个基本表时，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



5. 视图的作用

2. 视图使用户能以多种角度看待同一数据
 - 视图支持多用户同时以不同的方式对相同的数据进行查询
3. 视图对重构数据库提供了一定程度的逻辑独立性



5. 视图的作用

重构数据库最常见的是将一个基本表“垂直”地分成多个基本表

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)



5. 视图的作用

虽然数据库的逻辑结构改变了（变为**SX**和**SY**两个表），但通过建立一个视图**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**表的查询应用程序不必修改



5. 视图的作用

- 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。



5. 视图的作用

4. 视图能够对机密数据提供安全保护

- 针对不同用户定义不同视图，使机密数据不出现在不应看到这些数据的用户视图上



5. 视图的作用

5. 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，然后查询视图。

```
create view maxgmark
```

```
As
```

```
Select sno,max(gmark)
```

```
From sc
```

```
Group by sno
```



下课了。。。。

追
求



休息一会儿。。。。