SQL基本语法

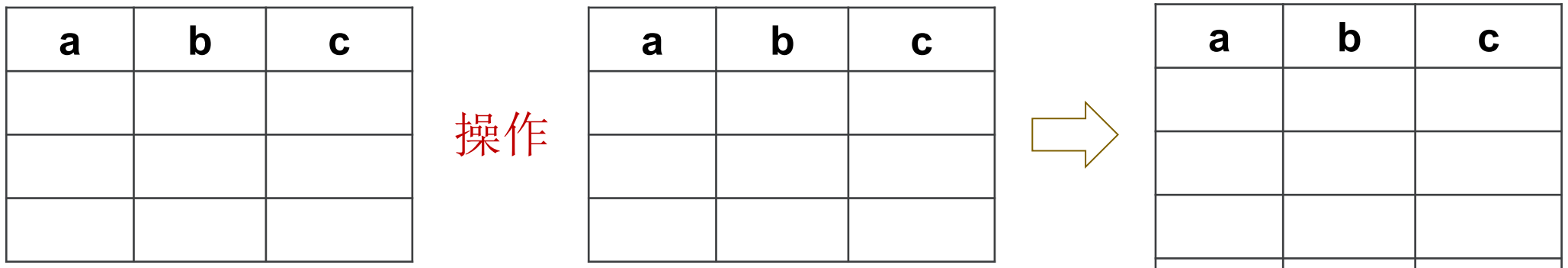
讲解人：李鸿岐

集合查询

数据查询SELECT

Combining Result Tables (UNION, INTERSECT, EXCEPT)

- In SQL, we can use the normal set operations of Union, Intersection, and Difference to combine the results of two or more queries into a single result table.
- The three set operators in the ISO standard are called UNION, INTERSECT, and EXCEPT



- 参加集合操作的各查询结果的列数必须相同;
- 对应项的数据类型也必须相同

数据查询SELECT

- Combining Result Tables (UNION, INTERSECT, EXCEPT)

```
(SELECT sNo
FROM SC
WHERE cNo='030101')
UNION
(SELECT sNo
FROM SC
WHERE cNo='030102');
```

```
(SELECT sNo
FROM SC
WHERE cNo='030101')
INTERSECT      EXCEPT
(SELECT sNo
FROM SC
WHERE cNo='030102');
```

数据查询SELECT

[案例] 查询计算机科学系的学生及年龄不大于19岁的学生。

```
SELECT *
FROM Student
WHERE Sdept= 'CS'
UNION
SELECT *
FROM Student
WHERE Sage<=19;
```

实际上是查询计算机科学系中年龄不大于19岁的学生

```
SELECT *
FROM Student
WHERE Sdept= 'CS' AND  Sage<=19;
```

- UNION：将多个查询结果合并起来时，系统自动去掉重复元组
- UNION ALL：将多个查询结果合并起来时，保留重复元组

数据操纵

数据操作语言DML

- Adding data to the database (INSERT)
  - Adds new rows of data to a table. There are two forms of INSERT statement.
  - The first allows a single row to be inserted into a named table.

```
INSERT INTO TableName [(columnList)]
VALUES (data ValueList)
```

  - The second form of the INSERT statement allows multiple rows to be copied from one or more tables to another.

```
INSERT INTO TableName [(columnList)]
 SELECT …
```

数据操作语言DML

INSERT INTO TableName [(columnList)]
VALUES (data ValueList)

- 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：插入的元组在其余属性列上取空值
- VALUES子句
  - 提供的值必须与INTO子句匹配
    - 值的个数
    - 值的类型

数据操作语言DML

案例

```
INSERT
INTO Student(Sno, Sname, Ssex, Sage, Sdept,)
VALUES ('170120', '陈冬', '男', 18, null, '01' );
```

```
INSERT INTO Student
VALUES ('170120', '陈冬', '男', 18, null, '01' );
```

```
INSERT INTO SC(sNo, cNo)
VALUES ('170120', '010101');
```

DBMS将在新插入记录的Score列上自动地赋空值。

类似于：

```
INSERT
INTO SC
VALUES ('170120','010101',NULL);
```

数据操作语言DML

- 案例

```
CREATE TABLE s_score(
  sNo    CHAR(6),
  sName  VARCHAR(20),
  avg    decimal(5,2));

INSERT INTO s_score(sNo,sName,avg)
  (SELECT s.sNo,s.sName,v.avgscore
   FROM student s,(SELECT sNo,avg(score) FROM sc GROUP BY sNo)
                  as v(sNo,avgscore)
   WHERE s.sNo=v.sNo);
```

数据操作语言DML

Adding data to the database (INSERT)

❖关系数据库管理系统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

数据操作语言DML

Modifying data in the database (UPDATE)

UPDATE TableName
SET columnName1=dataValue1[,columnName2=datavalue2…]
[WHERE searchCondition]

❖ 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

❖ 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

数据操作语言DML

- Modifying data in the database (UPDATE)

```
UPDATE Student
SET age=22
WHERE sNo='170101';
```

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

```
UPDATE SC
SET score=60
WHERE sNo=(SELECT sNo
                FROM Student
                WHERE sName='张敏')
  and cNo=(SELECT cNo
                FROM Course
                WHERE cName='数据库系统');
```

数据操作语言DML

Modifying data in the database (UPDATE)

❖关系数据库管理系统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

数据操作语言DML

Deleting data from the database (DELETE)

```
DELETE FROM TableName
[WHERE searchCondition]
```

- ❖ 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

```
DELETE FROM Student
WHERE sNo='070122';

DELETE FROM Student;

DELETE FROM SC
WHERE cNo=(SELECT cNo
                FROM Course
                WHERE cName='数据库系统');
```

数据操作语言DML

- 案例…

[案例] 删除学号为201215128的学生记录。

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

[案例] 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') ;
```

```
DELETE
FROM SC;
```

删除所有的学生选课记录

数据操作语言DML

- 案例…

高级SQL

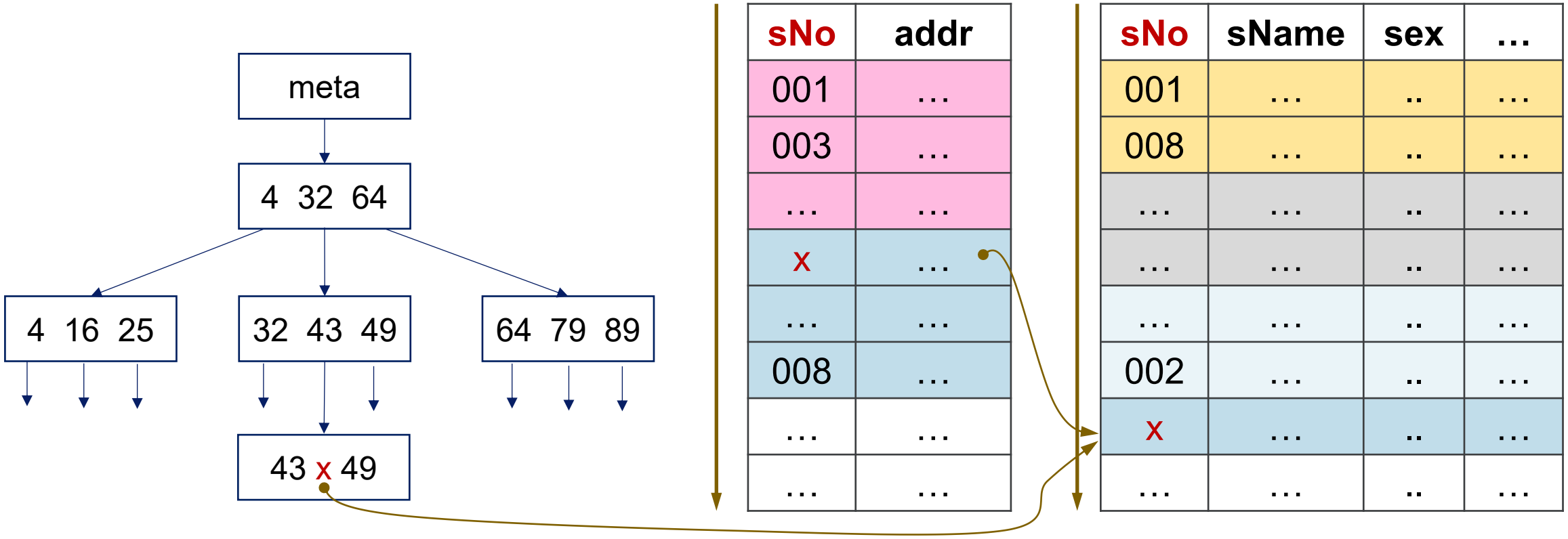
讲解人：李鸿岐

高级SQL

□ 索引(Index)

– An index is a structure that provides accelerated access to the rows of a table based on the values of one or more columns.

– We will discuss the index in detail in another topic.



高级SQL

索引(Index)

- 建立索引的目的：加快查询速度
- 关系数据库管理系统中常见索引：
  - 顺序文件上的索引
  - B+树索引
  - 散列（hash）索引
  - 位图索引
- 特点：
  - B+树索引具有动态平衡的优点
  - HASH索引具有查找速度快的特点

- 谁可以建立索引
  - 数据库管理员 或 表的属主（即建立表的人）
- 谁维护索引
  - 关系数据库管理系统DBMS自动完成
- 使用索引
  - 关系数据库管理系统自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引

高级SQL

Creating an Index (CREATE INDEX)

CREATE [UNIQUE] [CLUSTER] INDEX IndexName
ON TableName [USING method] (columnName[ASC | DESC][,…])

- <表名>：要建索引的基本表的名字
- 索引：可以建立在该表的一列或多列上，各列名之间用逗号分隔
- <次序>：指定索引值的排列次序，升序：**ASC**，降序：**DESC**。缺省值：**ASC**
- UNIQUE：此索引的每一个索引值只对应唯一的数据记录
- CLUSTER：表示要建立的索引是聚簇索引

高级SQL

Creating an UNIQUE Index

- 对于已含重复值的属性列不能建**UNIQUE**索引
- 对某个列建立**UNIQUE**索引后，插入新记录时**DBMS**会自动检查新记录在该列上是否取了重复值。这相当于增加了一个**UNIQUE**约束

Creating an CLUSTER Index

- 建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中记录的物理顺序一致

```
CREATE CLUSTER INDEX Stusname
ON Student(Sname);
```

■在**Student**表的**Sname**（姓名）列上建立一个聚簇索引，而且**Student**表中的记录将按照**Sname**值的升序存放

高级SQL

Creating an CLUSTER Index

- 在一个基本表上最多只能建立一个聚簇索引
- 聚簇索引的用途：对于某些类型的查询，可以提高查询效率
- 聚簇索引的适用范围
  - 很少对基表进行增删操作
  - 很少对其中的变长列进行修改操作
- 聚簇索引的不适用情形
  - 表记录太少
  - 经常插入、删除、修改的表
  - 数据平均分布的表字段

■在下列三种情况下，有必要建立簇索引：

(1)查询语句中采用该字段作为排序列

(2)需要返回局部范围的大量数据

(3)表格中某字段内容的重复性比较大

■例如，student表中dno一列有大量重复数据，当在dno列上建立了簇索引后，下面的连接查询速度会加快。

高级SQL

案例…

[案例] 为学生-课程数据库中的Student，Course，SC三个表建立索引。

Student表按学号升序建唯一索引，

Course表按课程号升序建唯一索引，

SC表按学号升序和课程号降序建唯一索引

```
CREATE UNIQUE INDEX  Stusno ON Student(Sno);

CREATE UNIQUE INDEX  Coucno ON Course(Cno);

CREATE UNIQUE INDEX  SCno ON SC(Sno ASC,Cno DESC);
```

CREATE INDEX sName-index ON Student(sName);
DROP INDEX sName-index;

ALTER an Index

❖ ALTER INDEX **<旧索引名>** RENAME TO **<新索引名>**

[案例] 将SC表的SCno索引名改为SCSno

ALTER INDEX SCno RENAME TO SCSno;

DROP an Index

❖ DROP INDEX **<索引名>;**

删除索引时，系统会从数据字典中删去有关该索引的描述。

[案例] 删除Student表的Stusname索引

DROP INDEX Stusname;

高级SQL

索引(Index)

- An index is a structure that provides accelerated access to the rows of a table based on the values of one or more columns.
- We will discuss the index in detail in another topic.

- 索引虽能够加速数据库查询，但需占用一定的存储空间
- 基本表更新时，索引要进行相应的维护，增加数据库负担

高级SQL

视图(VIEW)

- The dynamic result of one or more relational operations on the base relations to produce another relation.
- A view is a *virtual relation* that does not necessarily exist in the in the database but can be produced upon request by a particular user, at the time of request.
- The DBMS stores the definition of the view but not the data in the database.

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

高级SQL

sNo	sName	dNo	…
001	…	..	…
008	…	..	…
…	…	..	…
…	…	..	…
…	…	..	…
002	…	..	…
…	…	..	…
…	…	..	…

sNo	cNo	score
001	01	..
001	02	..
002	01	..
003	02	..
…	…	..

cNo	cName	credit	…
01	离散数学	..	…
02	…	..	…
…	…	..	…
…	…	..	…
…	…	..	…

选修离散数学所有学生名单及其成绩视图

cName	sName	score
离散数学	李兰	..
离散数学	张军	..
离散数学	赵敏	..
…	…	..
…	…	..

view_1

高级SQL

Creating a View (CREATE VIEW)

```
CREATE VIEW ViewName[(newColumnName[,…])]
AS subselect [WITH [CASCADED|LOCAL] CHECK OPTION];
```

– If WITH CHECK OPTION is specified, SQL ensures that if a row fails to satisfy the WHERE clause of the defining query of a view, it is not added to the underlying base table of the view.

Removing a View (DROP VIEW)

```
DROP VIEW ViewName [RESTRICT|CASCADE];
```

– If CASCAE is specified, DROP VIEW deletes all related dependent objects, in other words, all objects that reference the view.

高级SQL

案例

```
CREATE VIEW view_1
  AS SELECT s.sName, c.cName, sc.score
       FROM Student s, Course c, sc
       WHERE s.sNo=sc.sNo and c.cNo=sc.cNo
         and c.cName='离散数学';

CREATE VIEW view_2(cNo, numberOf)
  AS SELECT cNo, COUNT(*)
       FROM sc
       GROUP BY cNo;

DROP VIEW view_1;
```

```
SELECT sName, score
FROM view-1
WHERE score<60;

SELECT *
FROM view_2
WHERE numberOf>100;

DROP VIEW view_2;
```

高级SQL

组成视图的属性列名：全部省略或全部指定

– 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:

(1) 某个目标列是集函数或列表达式

(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

```
CREATE VIEW view_1
  AS SELECT s.sName, c.cName, sc.score
     FROM Student s, Course c, sc
     WHERE s.sNo=sc.sNo and c.cNo=sc.cNo
       and c.cName='离散数学';
```

```
CREATE VIEW view_2(cNo, numberOf)
  AS SELECT cNo, COUNT(*)
     FROM sc
     GROUP BY cNo;
```

```
CREATE  VIEW BT_S(Sno，Sname，Sbirth)
AS SELECT Sno，Sname，2005-Sage
FROM  Student
```

- 带表达式的视图
- 设置一些派生属性列, 也称为虚拟列--Sbirth
- 带表达式的视图必须明确定义组成视图的各个属性列名

高级SQL

案例

```
CREATE VIEW F_Student1(stdnum，name，sex，age，dept)
        AS  SELECT *
        FROM  Student
         WHERE Ssex='女';
```

- 将**Student**表中所有女生记录定义为一个视图
- **F_Student1**的属性列与**Student**的属性列一一对应
- 缺点：修改基表**Student**的结构后，**Student**表与**F_Student1**视图的映象关系被破坏，导致该视图不能正确工作
- 修改基本表**Student** 后，删除由该基本表导出的视图，重建相关视图

View Resolution

- 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

DBMS仅存储对视图的定义，并不存储相关的数据;

高级SQL

View Resolution

```
SELECT sName, score
FROM view_1
WHERE score<60;
```

view_1

```
SELECT s.sName, c.cName, sc.score
FROM Student s, Course c, sc
WHERE s.sNo=sc.sNo and c.cNo=sc.cNo
  and c.cName='离散数学';
```

```
SELECT s.sName, sc.score
FROM Student s, Course c, sc
WHERE s.sNo=sc.sNo and c.cNo=sc.cNo
  and c.cName='离散数学'
  and sc.score<60;
```

sName	score
..	50
..	55
..	48
..	..
…	..

高级SQL

View Resolution

[案例] 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;
```

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

视图消解转换后的查询语句为：

```
SELECT  Sno,Sage
FROM  Student
WHERE  Sdept= 'IS'  AND  Sage<20;
```

❖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。

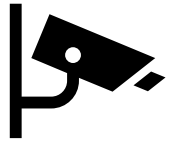
[案例]在S_G视图中查询平均成绩在90分以上的学生学号和平均成绩

```
SELECT *
FROM   S_G
WHERE  Gavg>=90;
```

S_G视图的子查询定义：

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

```
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;
```

❖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。

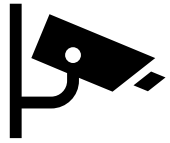
[案例]在S_G视图中查询平均成绩在90分以上的学生学号和平均成绩

```
SELECT *
FROM   S_G
WHERE  Gavg>=90;
```

S_G视图的子查询定义：

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

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



正确。

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

高级SQL

- View Updatability
  - All updates to a base table are immediately reflected in all views that encompass that base table.
  - Similarly, we may except that if a view is updated then the base table(s) will reflect that change.
  - Consider that if any view is updatable.

- 创建视图时，若取出了原有关系中的主码，则视图的更新可对应基本表的更新
- 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

高级SQL

- View Updatability
  - All updates to a base table are immediately reflected in all views that encompass that base table.
  - Similarly, we may except that if a view is updated then the base table(s) will reflect that change.
  - Consider that if any view is updatable.

[案例]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

对应基本表的更新

有意义地转换成对

View Updatability

❖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。

高级SQL

- 案例

```
CREATE VIEW IS_Student
AS SELECT sNo,sName,sex,age
  FROM student
  WHERE dNo IN(select dNo
               from department
               where dName='信息学院');
```

```
UPDATE IS_Student
SET sName='马成功'
WHERE sNo='070115';
```

↓

```
UPDATE Student
SET sName='马成功'
WHERE sNo='070115'
  AND dNo IN(…);
```

```
INSERT INTO IS_Student
VALUES('170199', '刘锋', '男', 20);
```

→

```
INSERT INTO Student
VALUES('170199', '刘锋', '男', 20, null);
```

```
DROP VIEW IS_Student;
```

```
DELETE FROM Student
WHERE sNo='170199';
```

高级SQL

❖ 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）

```
CREATE VIEW IS_Student
AS SELECT *
    FROM student
    WHERE dNo IN(SELECT dNo
          FROM department
          WHERE dName='信息学院')
WITH CHECK OPTION;
```

```
INSERT INTO IS_Student
VALUES('170199', '刘锋', '男', 20, '02');
```

```
ERROR:  new row violates check option for view "is_student"
DETAIL:  Failing row contains (170199, 刘锋, 男 , 20, 02).

********** 错误 **********

ERROR: new row violates check option for view "is_student"
SQL 状态: 44000
```

- 插入操作：**DBMS自动检查dno属性值是否为'01'**
  - 如果不是，则拒绝该插入操作

高级SQL

- Advantages of using view
  - Data independence
  - Currently
  - Improved security
  - Reduced complexity
  - Convenience
  - Customization
  - Data integrity
- Disadvantages of using view
  - Update restriction
  - Structure restriction
  - Performance

❖ 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

高级SQL

- Advantages of using view
  - Data independence
  - Currently
  - Improved security
  - Reduced complexity
  - Convenience
  - Customization
  - Data integrity
- Disadvantages of using view
  - Update restriction
  - Structure restriction
  - Performance

❖ 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
- 基于复杂嵌套查询的视图
- 含导出属性的视图

关于本讲内容

祝各位学习愉快！

感谢观看！

讲解人：李鸿岐