SQL 数据定义与更新语言

1 SQL 数据定义语言

2 SQL 数据更新语言

3 视图

SQL 的数据定义功能

- **DBMS 中可建立多个数据库；**
- 一个数据库包含了基本表、视图、索引以及约束等对象 .

操作对象	操作方式		
	创建	修改	删除
数据库	CREATE DATABASE	ALTER DATABASE	DROP DATABASE
表	CREATE TABLE	ALTER TABLE	DROP TABLE
索引	CREATE INDEX		DROP INDEX
视图	CREATE VIEW		DROP VIEW

例：Stu_Cou 数据库

主码 Sno

学生表 Student

学号 char(9) Sno	姓名 char(20) Sname	性别 char(2) Ssex	年龄 int Sage	系别 char(20) Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女		MA
[illegible]	张立	男		IS

主码 Cno

Cpno 是外码
被参照关系是 Course
被参照列是 Cno

课程表 Course

课程号 char(4)	课程名 char(20)	先行课 char(4) Cpno	学分 int Ccredit
		5	4
			2
		1	4
4	操作系统	6	3
		7	4
			2
		6	4

主码 (Sno,Cno)

选课表 SC

学 号 char(9) Sno	课程号 char(4) Cno	成绩 int Grade
201915121	1	56
201915121	2	
201915121	3	88
201915122	2	90
201915122	3	80

Cno 是外码
被参照关系是 Course
被参照列是 Cno

Sno 是外码
被参照关系是 Student
被参照列是 Sno

SQL 数据定义语言

1. 数据库的创建与删除
   - ① 创建数据库
   - ② 修改数据库
   - ③ 删除数据库

2. 基本表的定义删除与修改
   - ④ 定义表
   - ⑤ 修改表
   - ⑥ 删除表

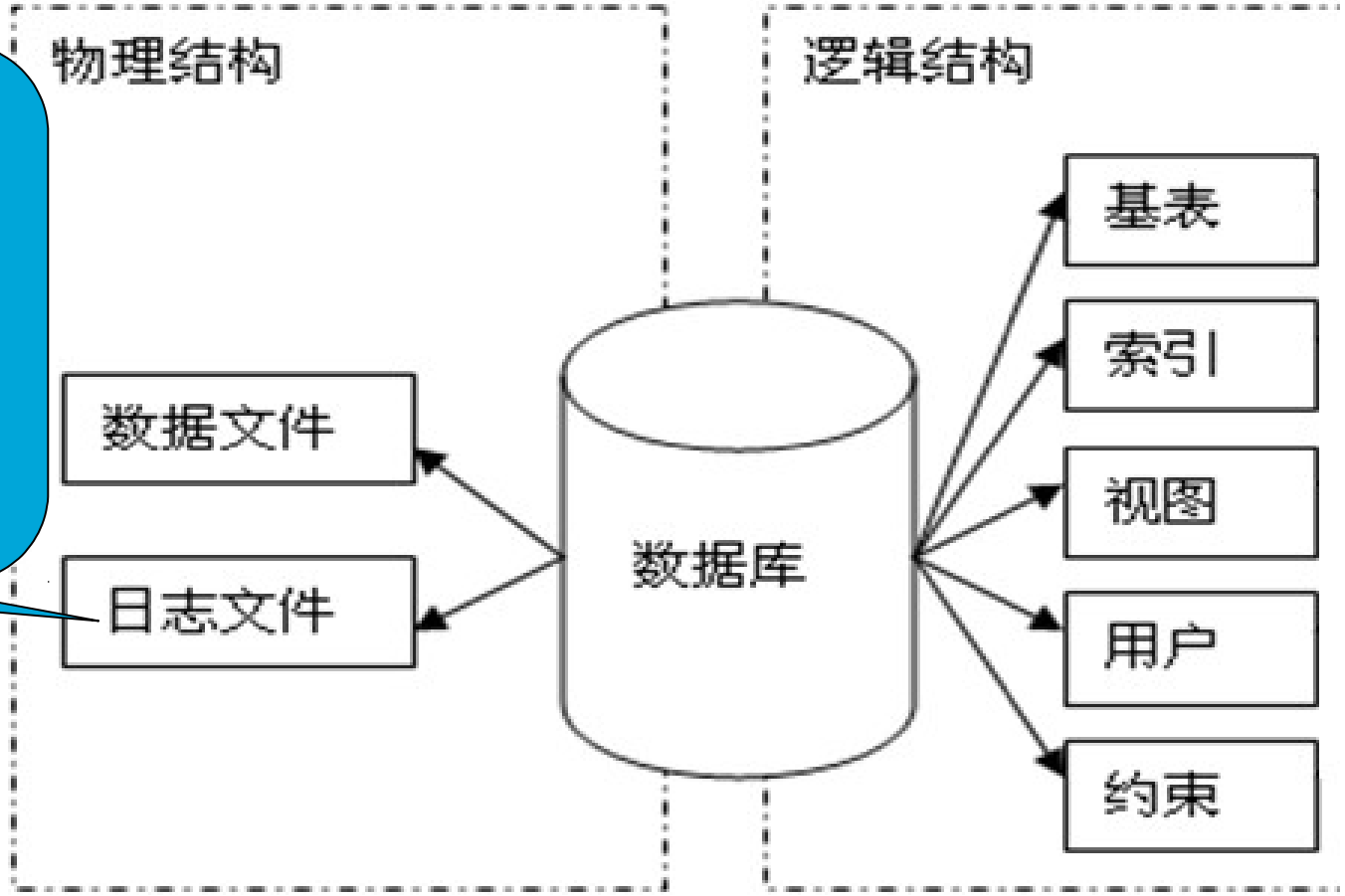
3. 索引的建立与删除
   - ⑦ 建立索引
   - ⑧ 删除索引

1. 数据库的创建与删除

✓ 数据库包含了基本表、视图、索引、约束等对象，在定义这些对象之前，必须首先定义数据库，然后在数据库中定义所有的对象.

数据库存储在文件中：

- 数据文件—存放数据库中对象的数据
- 日志文件—存放用于恢复数据库的冗余数据



① 创建数据库

例 1：新建数据库 Stu：指出数据文件和日志文件名称和位置．初始大小分别为 5MB 和 2MB，最大允许增加到 10MB 和 5MB，每次增量为 1MB 。

```
CREATE DATABASE Stu_Cou
ON ( NAME='Stu_dat',
     FILENAME='C:\Mydatabase\Stu_Cou.mdf',
     SIZE=5,
     MAXSIZE=10,
     FILEGROWTH=1 )
LOG ON
   ( NAME= 'Stu_log',
     FILENAME='C:\Mydatabase\Stu_Cou.ldf',
     SIZE=2,
     MAXSIZE=5,
     FILEGROWTH=1 )
```

① 数据库创建后，与该数据库相关的描述信息会保存到系统的数据字典中．

② 创建一个数据库时，仅创建了一个空壳．必须在这个空壳中创建对象(如表、索引、约束等).

② 修改数据库

```
ALTER DATABASE <databaseName>
{  ADD FILE <filespec> [, ...n] [TO FILEGROUP  <filegroupName>]
   | ADD LOG FILE <filespec> [, ... n]
   | REMOVE FILE <logicalFileName>
   | ADD FILEGROUP <filegroupName>
   | REMOVE FILEGROUP <filegroupName>
   | MODIFY FILE <filespec>
   | MODIFY FILEGROUP <filegroupName> <filegroupProperty>
}
```

例：修改 Stu_Cou 数据库，将逻辑文件名为 Stu_Cou 的磁盘文件的初始大小修改为 20M.

```
ALTER DATABASE Stu_Cou
MODIFY FILE ( NAME = TempHisDev1, SIZE=20MB)
```

③ 删除数据库

DROP DATABASE *<databaseName>*

例：删除数据库 Stu_Cou.

DROP DATABASE Stu_Cou

删除数据库时，系统会同时从系统的数据字典中将该数据库的描述一起删除 .
(有的 DBMS 会自动删除与数据库的物理文件)

2. 基本表的创建与删除

✓ 创建数据库后，就可在数据库中建立基本表．

① 定义基本表

CREATE TABLE < 表名 >
(< 列名 > < 数据类型 >[< 列级完整性约束条件 >]
[,< 列名 >< 数据类型 >[< 列级完整性约束条件 >]] …
[, < 表级完整性约束条件 >]);

注：表名后的括号中可以跟若干个列的定义，列与列之间用逗号分隔

SQL 中的基本数据类型是：

- 整型： **int (4B)** ， **smallint (2B)** ， **tinyint (1B)** ；
- 实型： **float** ， **real (4B)** ， **decimal(p, n)** ， **numeric(p, n)** ；
- 字符型： **char(n)** ， **varchar(n)** ， **text** ；
- **2** 进制型： **binary(n)** ， **varbinary(n)** ， **image** ；
- 逻辑型： **bit** ， 只能取 **0** 和 **1** ， 不允许为空；
- 货币型： **money (8B, 4** 位小数 **)** ， **small money (4B, 2** 位小数 **)** ；
- 时间型： **datetime (4B,** 从 **1753.1.1** 开始 **)** ，
  smalldatetime (4B, 从 **1900.1.1** 开始 **)**
- 其中： **image** 为存储图象的数据类型， **text** 存放大文本数据

例 1: 建立学生表 S
码，姓名取值唯一

```
CREATE TABLE Student
(    Sno    CHAR(9) PRIMARY KEY,
   Sname  CHAR(20) UNIQUE,
     Ssex    CHAR(2),
     Sage    INT,
     Sdept  CHAR(20)
)
```

注意：可以给约束命名：
Sno char(9) constraint c1 primary key,

主码

取值唯一

约束可以放在所有列定义之后：表级约束

```
CREATE TABLE Student
(     Sno   CHAR(9),
        Sname  CHAR(20),
      Ssex    CHAR(2),
        Sage   INT,
      Sdept  CHAR(20),
      primary key(Sno),
      unique(Sname)
)
```

学号 char(9) Sno	姓名 char(20) Sname	性 Ssex	Sage	
201915121	李勇	男	20	
201915122	刘晨	女	19	
201915123	王敏	女	18	
201915125	张立	男	19	

例 2: 建立课程表 Course(Cno,C... ...Ccredit)

主码 :Cno

```
CREATE TABLE  Course
(    Cno  CHAR(4) PRIMARY KEY,
     Cname  CHAR(20),
     Cpno    CHAR(4) ,
     Ccredit  INT,
     FOREIGN KEY (Cpno) REFERENCES Course(Cno)
)
```

Cpno 是外码
被参照表是
Course
被参照列是 Cno
表级约束

课程号 char(4) Cno	课程名 char(20) Cname	先行课 ch Cpno
1	数据库	5
2	数学	
3	信息系统	1
4	操作系统	6
5	数据结构	7
6	数据处理	
7	C 语言	6

```
CREATE TABLE  Course
( Cno  CHAR(4) PRIMARY KEY,
  Cname  CHAR(40),
  Cpno  CHAR(4) REFERENCES Course(Cno),
    Ccredit  INT
)
```

外码约束也可以放在列 Cpno 定义之后，作为列级约束.

例 3: 建立一个学生选课表

```
CREATE TABLE  SC
(   Sno  CHAR(9),
    Cno  CHAR(4),
    Grade   INT,
    PRIMARY KEY (Sno,Cno),
    FOREIGN KEY (Sno) REFERENCES Student(Sno),
    FOREIGN KEY (Cno) REFERENCES Course(Cno)
)
```

注意 1: 外码属性的类型必须与被参照表中主码属性类型一致！

注意 2: 主码是属性组时，则主码约束只能放在所有列定义之后，作为表级约束.

主码 :(Sno, Cno)

Sno 是外码
被参照关系是 Student
被参照列是 Sno

Cno 是外码
被参照关系是 Course
被参照列是 Cno

学 号 char(9) Sno	课程号 char(4) Cno	成绩 int Grade
201915121	1	56
201915121	2	44
201915121	3	88
201915122	2	90
201915122	3	80

② 修改基本表

```
ALTER TABLE < 表名 >
        [ ADD < 新列名 > < 数据类型 > [  完整性约束
          ] ]
        [ DROP COLUMN < 列名 >]
        [ DROP CONSTRAINT < 完整性约束名 > ]
        [ ALTER COLUMN< 列名 > < 数据类型 > ] ;
```

修改方式	使用的关键字
增加新列	**ADD**
增加新的完整性约束条件	
修改原有的列定义	**ALTER COLUMN**
删除列 / 列完整性约束	**DROP**

修改基本表—增加列

例 1. 向 Student 表增加 " 入学时间 " 列 , 类型为日期型 .

```
ALTER TABLE Student
ADD  S_entrance Datetime
```

注意 1: 如果基本表中原来已有数据，则新增加的列 S_entrance 的值一律为空值 .

注意 2: 如果基本表中原来已有数据 , 则新增的列 S_entrance 不能有 NOT NULL 约束 .

修改基本表—修改列定义

例 2. 将年龄的数据类型改为 SMALLINT 类型 .

```
ALTER TABLE Student
ALTER COLUMN Sage SMALLINT
```

注意：修改原有的列可能会破坏已有数据。

修改基本表—增加约束条件

例 3. 增加课程名称必须取唯一值的约束条件 .

```
ALTER TABLE Course
ADD UNIQUE(Cname)
```

也可以给增加的约束命名：

```
ALTER TABLE Course
ADD CONSTRAINT C1 UNIQUE(Cname)
```

修改基本表—删除约束条件

例 4. 删除课程名称必须取唯一值的约束 .

```
ALTER TABLE Course
DROP CONSTRAINT C1
```

修改基本表—删除列

例 5. 删除入学时间 S_entranc 列。

```
ALTER TABLE Student
DROP COLUMN S_entrance
```

③ 删除基本表

DROP TABLE < 表名 >[RESTRICT| CASCADE]

RESTRICT

- 如果有依赖基本表的对象，则表不能被删除 .

CASCADE

- 删除基本表的同时，相关依赖对象一起删除 .

T-SQL 中，语法为：
DROP TABLE < 表名 >

例 1: 删除 Student 表

DROP TABLE Student CASCADE

当执行此语句后，

- 基本表定义被删除 .
- 表中已有的数据被删除 .
- 表上建立的索引、视图、触发器被删除 .

问题：SQL Server 不支持 CASCADE，当执行语句：DROP TABLE Student 会发生什么？

SQLQuery1.sql - (l...39PP\cxleng (54))*

```
DROP TABLE Student
```

消息

消息 3726，级别 16，状态 1，第 1 行
无法删除对象 'Student'，因为该对象正由一个 FOREIGN KEY 约束引用。

3. 索引的建立与删除

Q1. 什么是索引？

- ✓ 索引是一个指向表中数据的指针
- ✓ 索引是在基本表的列上建立的
- ✓ 索引是关系数据库的内部实现技术，属于内模式的范畴 .

Q2. 建立索引的目的

- ✓ 提高数据查询的速度
- ✓ 保证数据的唯一性
- ✓ 加快表连接的速度

3. 索引的建立与删除

Q3. 谁可以建立索引

- ✓ DBA 或 表属主 (即建立表的人)
- ✓ DBMS 会自动建立索引 : PRIMARY KEY 列、 UNIQUE 列

Q4. 谁维护索引

- ✓ DBMS 自动完成

Q5. 谁使用索引

- ✓ DBMS 自动选择是否使用索引及使用哪些索引，用户不能选择使用索引

3. 索引的建立与删除

Q6. 建立索引的原则

① 基本表中元组数量越多，越有必要创建索引；查询频率高、实时性要求高的数据一定要建立索引.

② 对一个基本表不要建立过多索引.

③ 避免建立索引的情况：

- ✓ 包含太多重复值的列
- ✓ 查询中很少被引用的列
- ✓ 具有很多 NULL 值的列
- ✓ 需要经常插、删、改的列
- ✓ 记录较少的基本表
- ✓ 需要进行频繁大批量数据更新的基本表

① 建立索引

- ✓ **UNIQUE** 建立唯一索引 .
- ✓ 说明该索引的每一个索引值只对应唯一的数据记录 .

- ✓ **CLUSTERED**： 建立聚簇索引（即索引项的顺序与表中记录的物理顺序一致）
- ✓ **NONCLUSTERED**： 建立非聚簇索引（默认）

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED]
INDEX < 索引名 >
ON < 表名 >(< 列名 >[< 次序 >][,< 列名 >[< 次序 >] ]…);
```

次序指定索引值的排列顺序 .

- ■ **ASC(默认) 为升序 .**

DESC 为降序 .

例 1: 为学生 - 课程数据库的三个表建立索引 , 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)
```

注意 1：对于含重复值的属性列不能建唯一索引 .

注意 2：对某个列建立唯一索引后，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，相当于增加了一个 UNIQUE 约束 .

例 2: 在 Student 表的 Sname 列建立一个聚簇索引 .

```
CREATE CLUSTERED INDEX Stusname
ON   Student(Sname)
```

① 建立聚簇索引后，基本表中数据也需要按指定的聚簇属性值的升序或降序存放，即聚簇索引的索引项顺序与表中记录的物理顺序一致 .

② 在最经常查询的列上建立聚簇索引可以提高查询效率 .

③ 一个基本表上最多只能建立一个聚簇索引 .

④ 经常更新的列不宜建立聚簇索引 .

② 删除索引

DROP INDEX < 索引名 > ON < 表名 >

删除索引时，系统会从数据字典中删去有关该索引的描述 .

一次可以删除多个索引，索引名之间用逗号隔开 .

例：删除 Student 表的索引 Stusname.

```
DROP INDEX Stusname ON
Student
```

SQL 数据定义与更新语言

1 SQL 数据定义语言

2 SQL 数据更新语言

3 视图

SQL 数据更新语言

- 插入数据
- 修改数据
- 删除数据

（1）插入单个元组

② 没有指定的属性列，新元组在这些列上取空值
但表定义中说明 NOT NULL 的属性列不能取空值 .

INSERT INTO < 表名 > [(< 属性列 1>,< 列 2 >…)]

VALUES (< 常量 1> [,< 常量 2>]　…)

① 属性列的顺序可与表定义中的顺序不一致；

③ 常量值的个数及类型必须与 INTO 子句匹配 .

不指定属性列

例 1: 已知 Student 表如右所示，现需增加一个学生信息:
(姓名 : 陈冬、学号 :201915128 、系别 : IS 、性别 : 男、年龄 :18 岁) .

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	20	MA
201915125	张立	男	19	IS

INSERT
INTO Student ✗
VALUES (' 陈冬 ','201915128','IS', ' 男 ',1

方法 1: 不指定属性列
INSERT INTO Student √
VALUES ('201915128',' 陈冬 ',' 男 ', 18, 'IS')

INTO 后只有表名，说明新增元组要在表的所有属性列上指定值，且属性列的次序与表定义中属性列的次序一致。

指定属性列

例 1: 已知 Student 表如右所示，现需增加一个学生信息:
(姓名 : 陈冬、学号 :201915128 、系别 : IS 、性别 : 男、年龄 :18 岁) .

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	20	MA
	张立	男	19	IS

方法 2: 指定属性列

```
INSERT
INTO  Student(Sname, Sno, Sdept, Ssex, Sage)
VALUES (' 陈冬 ','201915128','IS', ' 男 ',18)
```

√

INTO 后属性列的顺序可与表定义中的顺序不一致 .

常量值的个数及类型必须与 INTO 后属性列匹配 .

指定属性列

例 2: 向表 SC(Sno, Cno, Grade) 中插入一条选课记录 :('201915128','1')

```
INSERT
INTO SC
VALUES ('201915128', '1')
```

×

请思考：SC 表中 Sno 、 Cno 是外码，如果 Student 表中没有 20195128 的学生记录，那么该插入语句能成功执行吗?

方法 1:

```
INSERT
INTO SC (Sno, Cno)
VALUES ('201915128', '1')
```

√

新元组在没有指定的属性列上取空值．即 Grade 列上取值为空．

NULL 值的插入

例 2: 向表 SC(Sno, Cno, Grade) 中插入一条选课记录 :('201915128','1')

```
INSERT
INTO SC
VALUES ('201915128', '1')
```

×

方法 2:

```
INSERT
INTO SC
VALUES ('201915128','1', NULL)
```

√

NULL 值的插入

例 3: 表 SC(Sno, Cno, Grade) 中主码为 (Sno, Cno)，若执行下列 SQL 语句：

```
INSERT
INTO SC
VALUES ('201915128', NULL, 85)
```

错误！属性 Cno 是主属性不能为空值！

结论：如果一个列被指定为 NOT NULL，那么使用 INSERT 时，必须为该列插入一个值，否则此次 INSERT 操作将被取消并提示错误信息。

唯一值的插入

例 4: 已知 Student 表如右所示

.

且 Sname 被指定 UNIQUE 约束，

则当执行下面的 SQL 语句：

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	20	MA
201915125	张立	男	19	IS

INSERT INTO Student

VALUES ('201915128',' 李勇 ',' 男 ', 18, 'IS') ✗

错误！这是因为违背了唯一性约束 .

结论：若一个属性列被定义了 UNIQUE 约束，那么执行 INSERT 时，系统自动检查表中是否已经存在新增的列值。
若存在则此 INSERT 操作将被拒绝执行，并提示错误信息 .

(2) 插入子查询结果

注意 2：在 INSERT INTO 语句中，列的数目及类型必须等于从 SELECT 语句返回的列的数目及类型。

INSERT INTO < 表名 > [(< 属性列 1>,< 列 2 >…)]

SELECT… FROM…WHERE…

子查询不仅可以嵌套在 SELECT 语句中，还可用在 INSERT 、 DELETE 、 UPDATE 、 CREATE VIEW 语句中 .

注意 1: SELECT 语句不能从正在被插入的表中选择数据

(2) 插入子查询结果

例 1：求每一个系的学生平均年龄，并把结果存入数据库中 .

第一步：建表 Dept_age.

```
CREATE  TABLE  Dept_age
(       Dept  CHAR(20),
        Avgage INT
)
```

问题：有没有更简洁的方法实现该操作？

第二步：插入数据

```
INSERT  INTO  Dept_age
        SELECT  Sdept, AVG(Sage)
    FROM  Student
    GROUP BY Sdept
```

INSERT…SELECT 可实现将一个表或多个表的数据复制到新表中，但是需要事先创建好新表 .

(3) 表中数据的复制

SELECT < 列 1>,< 列 2 >,…

INTO 新表

FROM 表 1, 表 2,…

WHERE …

SELECT…INTO 语句可以将一个或多个表的数据复制到新表中，且不需要事先创建新表。

SQL Server 支持该语句！

例 1: 求每一个系的学生平均年龄，并把结果存入数据库中 .

```
SELECT  Sdept AS Dept, AVG(Sage) AS Avgage
INTO  Dept_age
FROM  Student
GROUP BY Sdept
```

SQL 数据更新语言

- 插入数据
- 修改数据
- 删除数据

2. 修改数据

✓功能：修改指定表中满足 WHERE 子句条件的元组。

```
UPDATE <表名>
SET <列名>=<表达式>[,<列名>=<表达式>]...
[WHERE <条件>];
```

✓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。

修改某一个元组的值

例 1：将学生 201915121 的年龄改为 22 岁 .

```
UPDATE  Student
SET Sage=22
WHERE  Sno='201915121'
```

修改某一个元组的值

例 2： 已知 SC 表中的 Sno 是外码，参照 Student 表的 Sno 列：

Student(Sno,Sname,Ssex,Sage,Sdept)

SC(Sno,Cno,Grade)

问题：若 SC 表中存在李勇的选课记录，现需修改 Student 表中“李勇”的学号为“ 201915001”，执行下列 SQL 语句，正确吗?

```
UPDATE Student
SET Sno= '201915001'
WHERE Sname=' 李勇 '
```

请同学们思考为什么会出错？有什么解决的方法？

修改多个元组的值

例 3: 将所有学生的年龄增加 1 岁 .

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

没有 **WHERE** 子句，即修改 **Student** 表中的所有元组 .

修改多列数据

例 4: 将李勇的年龄增加 1 岁、系名修改为 'IS'.

```
UPDATE Student
SET Sage= Sage+1, Sdept='IS'
WHERE Sname=' 李勇 '
```

在 SET 子句后可以给出需要修改的多列数据，用逗号隔开 .

修改某列数据为空

例 5: 将选修了 1 号课程的成绩设为空 .

```
UPDATE SC
SET Grade=NULL
WHERE Cno='1'
```

注意：不能将主属性或 NOT NULL 约束的列值修改为 NULL.

思考问题：

1. 某些属性上的取值为空，是怎样产生的？
2. 如何判断某一个属性的值是否为空？

带子查询的修改语句

例 6: 将 CS 系全体学生的成绩置零。

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

根据 Student 表的值来修改 SC 表中的数据 .

带子查询的修改语句

例 7: 将低于总成绩平均分的成绩提高 5% 。

```
UPDATE SC
SET Grade=Grade*1.05
WHERE Grade<
(    SELECT AVG(Grade)
     FROM SC
)
```

SQL 数据更新语言

- 插入数据
- 修改数据
- 删除数据

删除数据

DELETE

FROM　< 表名 >

[WHERE < 条件 >] ;

注意 1: DELETE 后没有列名 !

注意 4： 同 INSERT 和 UPDATE 语句一样，注意参照完整性问题 .

注意 2:DELETE 语句不能删除单个字段的值，只能删除整行数据。

问题：怎样删除单个字段的值？

用 UPDATE 语句将该字段修改为 NULL.

注意 3:DELETE 仅删除表中数据，表的定义仍在数据字典中。

问题：怎样删除表?

DROP TABLE 语句删除表

删除某一个元组的值

例 1: 删除学号为 201915128 的学生记录 .

```
DELETE
 FROM Student
 WHERE Sno= '201915128'
```

问题：如果 SC 表的 Sno 参照 Student 表的 Sno，且在 SC 表中有 201915128 学生的选课记录，那么该语句能成功执行吗？

删除所有元组

例 2：删除所有的学生选课记录。

```
DELETE
FROM SC
```

带子查询的删除语句

例 3: 删除 CS 系所有学生的选课记录。

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

根据 Student 表中 Sdept 的值来删除 SC 表中的数据。

SQL 数据定义与更新语言

1 SQL 数据定义语言

2 SQL 数据更新语言

3 视图

什么是视图 ？

- ✓视图是从 *一个或多个基本表或视图* 中导出的表 .
- ✓视图是一张虚表，数据库中只存放视图的定义，不存放视图对应的数据 .

注意：视图所对应的数据并不实际存储在数据库中，而是存储在视图所引用的基本表中 .

- ✓当对视图中的数据进行修改时，相应的基本表的数据发生变化；同样地，若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

问题 1：若基表中的数据发生变化，从视图中查询出的数据会不会受到影响也发生变化？

问题 2：当对视图中的数据进行修改时，相应基本表的数据会不会受影响？

对视图进行的操作

视图定义后，就可以对其进行如下操作：

查询

删除

注意：这里删除的是视图

受限更新（即数据的增加 / 删除 / 修改）

定义基于该视图的新视图

注意：这里删除的是视图所对应基本表中的数据。

视图

- 视图的创建与删除
- 查询视图
- 更新视图
- 视图的作用

一、定义视图

CREATE VIEW < 视图名 > [(< 列名 > [,< 列名 >]…)]

AS < 子查询 >

[WITH CHECK OPTION]

注意 1: 组成视图的属性列名：全部省略或全部指定 .

注意 2： WITH CHECK OPTION 短语表示对视图进行 UPDATE 、 INSERT 和 DELETE 操作时要保证满足子查询中的条件表达式 .

1. 行列子集视图

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，则称这类视图为行列子集视图 .

例 1: 建立 IS 系学生的视图 IS_Stud

```
CREATE VIEW IS_Student
AS
SELECT *
FROM    Student
WHERE  Sdept= 'IS'
```

未指定视图的列名，则视图的列名隐含为 **SELECT** 子句中的列名组成 .

注意： **DBMS** 执行该语句时，只是把视图的定义存放在**数据字典**中，并不执行其中的 **SELECT** 子句。
在对视图查询时，按照该语句的定义从基本表中将数据查询出来。

2. WITH CHECK OPTION 的视图

例 2. 建立 IS 系学生的视图 IS_Student，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 *
FROM  Student
WHERE  Sdept= 'IS'
WITH CHECK OPTION
```

对 IS_Student 视图进行：

① 修改操作：自动加上 Sdept= 'IS' 的条件

② 删除操作：自动加上 Sdept= 'IS' 的条件

③ 插入操作：自动检查 Sdept 是否

3. 基于多个基本表的视图

例 3: 建立 IS 系选修了 1 号课程的学生视图 IS_S1 。

```
CREATE VIEW IS_S1(Sno, Sname, Grade)
AS
SELECT Student.Sno, Sname,Grade
FROM  Student, SC
WHERE Student.Sno=SC.Sno AND Sdept= 'IS' AND Cno= '1'
```

结论 1: 当多表连接时选出了几个同名列作为视图的字段时，需明确指定组成视图所有列名 .

或者通过定义别名实现 :

```
CREATE VIEW IS_S1
AS     SELECT Student.Sno as Sno, Sname, Grade
       FROM  Student, SC
       WHERE Student.Sno=SC.Sno AND Sdept= 'IS' AND Cno= '1'
```

4. 基于视图的视图

例 3: 建立 IS 系选修了 1 号课程的学生视图 IS_S1 。

若已经建立了 IS 系的学生视图 IS_Student， 是否可以在视图 IS_Student 的基础上建立?

```
CREATE VIEW IS_S1(Sno, Sname, Grade)
AS
SELECT IS_Student.Sno, Sname, Grade
FROM  IS_Student, SC
WHERE IS_Student.Sno=SC.Sno AND Cno= '1'
```

4. 基于视图的视图

例 4： 在视图 IS_S1 上建立 IS 系选修了 1 号课程且成绩在 90 分以上的学生的视图 IS_S2 。

```
CREATE VIEW IS_S2
AS
    SELECT Sno, Sname, Grade
    FROM  IS_S1
    WHERE Grade>=90
```

结论 2：视图可以：

① 从一个基本表导出；

② 从多个基本表导出；

③ 从一个已定义视图导出；

④ 从多个已定义视图导出；

⑤ 从视图和基本表中导出 .

5. 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, YEAR(GETDATE())-Sage
        FROM  Student
```

结论 3: 当 SELECT 后的目标列是表达式时，应该明确指定组成视图所有列名 .

6. 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 .

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

结论 4: 当 SELECT 后的某个目标列是聚集函数时，应该明确指定组成视图所有列名 .

或者通过定义别名实现：

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

二、删除视图

DROP VIEW <视图名> [CASCADE]

功能：从数据字典中删除视图定义 .

CASCADE：如果在该视图上还定义了其他视图，则使用 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 .

注意！SQL Server 中不支持 CASCADE 。如果要删除一个视图，应在执行 DROP VIEW 语句之前先把在该视图上定义的其他对象删除

视图

- 视图的创建与删除
- 查询视图
- 更新视图
- 视图的作用

查询视图

- ✓ 视图定义后，对视图的查询与对基本表查询相同 .
- ✓ RDBMS 实现视图查询的方法 :

  视图消解法（ View Resolution ）

  ① 进行有效性检查

  ② 转换成等价的对基本表的查询

  ③ 执行修正后的查询

查询视图

例 1: 若已有 IS 系学生的视图定义：

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

在 IS_Student 视图中找出年龄小于 20 岁的学生的学号和年龄。

```
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

例 2: 若已有 IS 系学生的视图定义 :

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

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'
```

视图消解转换后的查询语句为 :

```
SELECT  Student.Sno, Sname
 FROM  Student, SC
 WHERE  Sdept= 'IS'  AND Student.Sno=SC.Sno AND  SC.Cno= '1'
```

视图

- 视图的创建与删除
- 查询视图
- 更新视图
- 视图的作用

更新视图

- ✓视图是一张虚表，对视图的更新操作转换为对基本表的更新 .
- ✓DBMS 实现视图更新的方法：视图消解法 .
- ✓定义视图时指定 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 .

1. 行列子集视图的更新

例 1: 若已有 CS 系学生的视图定义：

```
CREATE VIEW CS_Student
AS
SELECT Sno, Sname, Sage
FROM    Student
WHERE  Sdept= 'CS'
```

请分析下列 SQL 语句执行后

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	18	MA
201915125	张立	男	19	IS

视图消解转换后的语句为：

UPDATE Student SET Sname= ' 刘辰 '
WHERE Sage=19 AND Sdept='CS'

实际上是对基本表中数据的修改！

```
SELECT * FROM Student
SELECT * FROM CS_Student
UPDATE  CS_Student SET  Sname= ' 刘辰 '  WHERE  Sage=19
SELECT * FROM Student
SELECT * FROM CS_Student
```

结论

结论 1: 行列子集视图是可以更新的 (即插入、删除、修改) 。

结论 2: 对视图的更新操作最终要转换为对基本表的更新 .

2. with check option 视图的更新

例 2: 若有视图定义 :

```
CREATE VIEW CS_Stu
AS
     SELECT  Sno , Sname , Sage, Sde
      FROM Student
     WHERE  Sdept ='CS'
```

若执行下列插入语句 :

```
INSERT
INTO CS_Stu
VALUES ('201915129', ' 赵新 ', 20, 'MA')
```

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	20	MA
201915125	张立	男	20	IS

会看到在 Student 表中有该条记录吗 ?会

该语句会被执行吗？会

若执行： Select * from cs_stu 查询结果中有该记录吗 ?

没有该记录，因为该记录的 Sdept 字段值不等于 'CS'

2. with check option 视图的更新

例 3: 视图定义如下：

```
CREATE VIEW CS_Stu1
AS  SELECT  Sno , Sname , Sage, Sdept
      FROM Student
      WHERE  Sdept ='CS'
      WITH CHECK OPTION
```

若执行下列插入语句：

```
INSERT
INTO CS_Stu1
VALUES ('201915129', ' 赵新 ', 20, 'MA')
```

Sno	Sname	Ssex	Sage	Sdept
201915121	李勇	男	20	CS
201915122	刘晨	女	19	CS
201915123	王敏	女	20	MA
201915125	张立	男	20	IS

该语句会被执行吗？不会

视图定义时有： **WITH CHECK OPTION** 子句，表示对视图执行 **update** 、 **insert** 、 **delete** 操作时，要保证修改、插入或删除的行必须满足视图定义中 **WHERE** 后的条件 .

结论

结论 3：定义视图有 **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。

3. 基于多个基本表的视图的更新

例 5: 视图 ms 的定义如下 :

```
create view ms(Sno, Sname, Cno, Grade)
as select Student.Sno, Sname, Cno, Grade
    from Student, SC
    where Student.Sno=SC.Sno AND Sdept= 'CS'
```

若执行下列删除语句 :

```
delete from ms
where Sno='201915122'
```

在 SQL Server 2008 中执行该语句失

该语句会被执行吗？

SQL Server 规定：当视图是由多个基本表导出的，不能使用 DELETE 语句删除数据 .

3. 基于多个基本表的视图的更新

例 6: 视图 ms 的定义如下 :

```
create view ms(Sno, Sname, Cno, Grade)
as select Student.Sno, Sname, Cno, Grade
     from Student, SC
     where Student.Sno=SC.Sno AND Sdept= 'CS'
```

若执行下列插入语句 :

```
insert into ms
values('201915155',' 杨阳 ','1',80)
```

该语句会被执行吗？

在 SQL Server 2008 中执行该语句失败！

SQL Server 规定：当视图是由多个基本表导出，若使用 INSERT 插入记录，要求插入的列必须属于同一个基本表。

3. 基于多个基本表的视图的更新

例 7: 视图 ms 的定义如下：

```
create view ms(Sno, Sname, Cno, Grade)
as select Student.Sno, Sname, Cno, Grade
    from Student, SC
    where Student.Sno=SC.Sno AND Sdept= 'CS'
```

将 201915121 学生的姓名修改为‘张帆’的语句为：

```
update ms
set Sname=' 张帆 '
where Sno='201915121'
```

在 SQL Server 中执行该语句

该语句会被执行吗？

SQL Server 规定：当视图是由多个基本表导出，若使用 UPDATE 修改记录，要求一次修改该视图只能变动一个基本表

结论

结论 4: 若视图是由多个基本表导出，则更新受限 .

4. 分组视图的更新

例 8: 若 S_G 视图定义如下 :

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

将视图中学号为‘ 201915121’ 学生的平均成绩修改为 90 分 :

```
UPDATE  S_G
SET Gavg=90
WHERE  Sno= '201915121'
```

在 SQL Server 中执行该语句失败！

该语句会被执行吗？

结论

结论 5：若视图定义中包含聚合函数，则该视图不能更新 .

① 并不是所有的视图都允许被更新！

② 不同的 **DBMS** 对视图更新的限制规定也不相同！

视图

- 视图的创建与删除
- 查询视图
- 更新视图
- 视图的作用

视图的作用

1. 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

2. 视图使用户能以多种角度看待同一数据

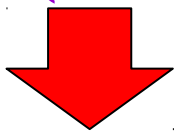
视图机制能使不同用户以不同方式看待同一数据，适应数据库共享的需要

视图的作用

3. 视图对重构数据库提供了一定程度的逻辑独立性

例：当数据库的逻辑结构发生变化时：

Student(Sno, Sname, Ssex, Sage, Sdept)



SX(Sno, Sname, Sage)

SY(Sno, Ssex, Sdept)

为了使对原 Student 表的查询程序不必修改，即用户的外模式不变，可以通过建立一个视图 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
```

视图的作用

4. 视图能够对机密数据提供安全保护

① 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 .

如：规定学生只有查看自己的成绩的权限 .

② 通过 **WITH CHECK OPTION** 对关键数据定义操作限制 .

视图的作用

5. 适当的利用视图可以更清晰的表达查询

例 : 查询每个同学获得最高成绩的课程号 .

① 建立视图 VMGRADE 表示每个学生的最高成绩 .

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

② 从 VMGRADE 和表 SC 中查找每个学生最高成绩对应的课程号 .

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