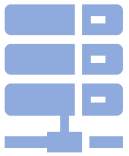


第三章 SQL语言

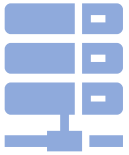
第三章 SQL语言

1. 查询语言
2. DDL语言
3. DML语言
4. DCL语言
4. 视图

定义数据库模式

- 数据库模式（schema）主要由数据库中关系（relation）的声明构成。我们首先介绍关系的定义
- 数据库模式也包括视图（view）、索引（index）、触发器（trigger）等对象，这些对象我们在稍后课程中介绍
- 对数据库模式的定义通过SQL语言中的数据定义语言（Data Definition Language）部分来完成
- DDL语言完成数据库对象的创建、删除和修改

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



定义基本表

■ 定义语句格式

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>]] …

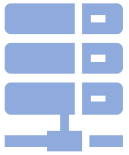
[，<表级完整性约束条件>] ）；

✓<表名>：所要定义的基本表的名字

✓<列名>：组成该表的各个属性（列）

✓<列级完整性约束条件>：涉及相应属性列的完整性约束条件

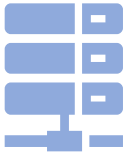
✓<表级完整性约束条件>：涉及一个或多个属性列的完整性约束条件



定义基本表

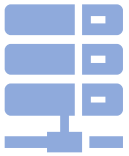
[例1] 建立一个“学生”表Student，它由学号Sno、姓名Sname、性别Ssex、年龄Sage、所在系Sdept五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREATE TABLE Student
        (Sno      CHAR(5)  NOT NULL  UNIQUE,
         Sname  CHAR(20)  UNIQUE,
         Ssex      CHAR(1) ,
         Sage      INT,
         Sdept    VARCHAR(15));
```



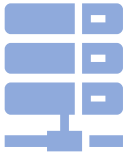
定义基本表

- 常见的数据类型
  - INT or INTEGER
  - REAL or FLOAT
  - CHAR(n) = 定长字符串
  - VARCHAR(n) = 变长字符串，n是字符串最大长度
  - DATE, TIME, and DATETIME



定义基本表

- 常用完整性约束
  - 主键约束：　PRIMARY KEY
  - 唯一性约束：UNIQUE
  - 非空值约束：NOT NULL
  - 参照完整性约束：　REFERENCES



定义基本表

[例2] 建立一个“学生选课”表SC，它由学号Sno、课程号Cno，修课成绩Grade组成，其中(Sno, Cno)为主键。

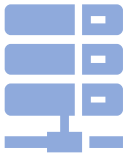
```
CREATE TABLE SC(
        Sno CHAR(5) ,
        Cno CHAR(3) ,
        Grade   int,
        Primary key (Sno, Cno));
```

删除基本表

DROP TABLE <表名>;

[例3] 删除Student表

DROP TABLE Student ;



修改基本表

ALTER TABLE <表名>

[ADD <新列名> <数据类型> [完整性约束]]

[DROP <完整性约束名>]

[MODIFY <列名> <数据类型>];

<表名>：要修改的基本表

ADD子句：增加新列和新的完整性约束条件

DROP子句：删除指定的列或完整性约束条件

MODIFY子句：用于修改列名和数据类型

修改基本表

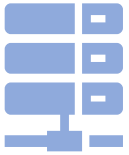
[例4]　向Student表增加“入学时间”列，其数据类型为日期型。

ALTER TABLE Student ADD Scome DATE;

✓不论基本表中原来是否已有数据，新增加的列一律为空值。

[例5] 删除属性列

ALTER TABLE Student Drop Sage;



修改基本表

[例6] 将年龄的数据类型改为半字长整数。

```
ALTER TABLE Student MODIFY Sage SMALLINT;
```

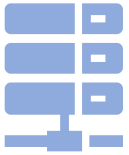
✓注：修改原有的列定义有可能会破坏已有数据

[例7] 删除学生姓名必须取唯一值的约束。

```
ALTER TABLE Student DROP UNIQUE(Sname);
```

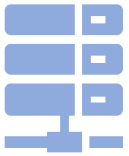
第三章 SQL语言

1. 查询语言
2. DDL语言
3. DML语言
4. DCL语言
4. 视图



数据更新

- Data Modification Language，数据更新语言
- 数据更新语句不返回关系表，而是会改变数据库的内容
- 三种数据更新类型
  - Insert 插入一条或多条元组
  - Delete 删除一条或多条元组
  - Update 更新现有一条或多条元组中的值



数据更新

- 插入数据
  - 插入单条元组
  - 插入子查询结果

插入单条元组语句格式

INSERT
INTO <表名> [(<属性列1>[，<属性列2 >...)]
VALUES (<常量1> [，<常量2>] ...)

数据更新

- 插入数据
  - 插入单条元组

[例7] 将一个新学生记录

(学号：95020；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

```
INSERT
  INTO Student
  VALUES ('95020'，'陈冬'，'男'，'IS'，18)；
```

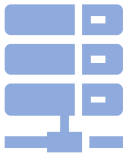
数据更新

- 插入数据
  - 插入单条元组

[例8] 插入一条选课记录('95020'，'1 ')。

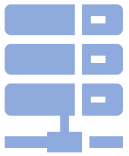
```
INSERT
 INTO SC(Sno，Cno)
 VALUES (' 95020 '，' 1 ');
```

新插入的记录在Grade列上取空值



数据更新

- 插入数据
  - 插入单条元组
    - 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



数据更新

- 插入数据
  - 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;

    功能

    将子查询结果插入指定表中

数据更新

- 插入数据
  - 插入子查询结果

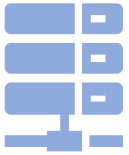
[例9] 对每一个系，求学生的平均年龄，并把结果存入数据库。

第一步：建表

```
CREATE TABLE Deptage
    (Sdept CHAR(15), Avgage SMALLINT)
```

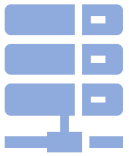
第二步：插入数据

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



数据更新

- 插入数据
  - 插入子查询结果
    - ✓INTO子句(与插入单条元组类似)
      - 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
      - 指定部分属性列：插入的元组在其余属性列上取空值
    - ✓ 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



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

  - 三种修改方式

    修改某一个元组的值

    修改多个元组的值

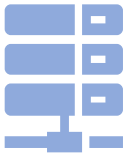
    带子查询的修改语句

数据更新

- 修改数据
  - 修改某一个元组的值

[例10] 将学生95001的年龄改为22岁。

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 95001 ';
```

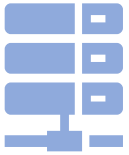


数据更新

- 修改数据
  - 修改某多个元组的值

[例11] 将所有学生的年龄增加1岁。

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

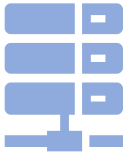


数据更新

- 修改数据
  - 带子查询的修改语句

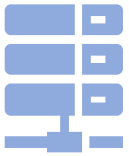
[例12] 将计算机科学系全体学生的成绩置零。

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



数据更新

- 修改数据
  - 带子查询的修改语句
    - 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



数据更新

■ 删除数据

● 语法格式

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

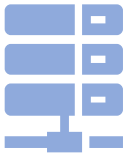
　　缺省表示要修改表中的所有元组

● 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

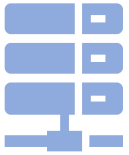


数据更新

- 删除数据
  - 删除某一个元组的值

[例13] 删除学号为95019的学生记录。

```
DELETE
FROM Student
WHERE Sno='95019';
```



数据更新

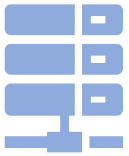
- 删除数据
  - 删除某多个元组的值

[例14] 删除2号课程的所有选课记录。

```
DELETE
FROM SC;
WHERE Cno='2';
```

[例15] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```



数据更新

- 删除数据
  - 带子查询的删除语句

[例16] 删除计算机科学系所有学生的选课记录。

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

第三章 SQL语言

1. 查询语言
2. DDL语言
3. DML语言
4. DCL语言
4. 视图

DCL语言

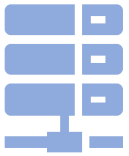
Data Control Language， 数据控制语言

- 授予或回收访问数据库的某种特权
- 控制数据库操纵事务发生的时间及效果
- 对数据库实行监视等

在后面课程中讨论学习

第三章 SQL语言

1. 查询语言
2. DDL语言
3. DML语言
4. DCL语言
4. 视图

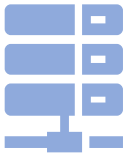


视图

- 视图是一种虚表（virtual table），是从一个或几个基本表（或视图）导出的表
  - 视图其实是在数据字典中存储的一条Select 语句

    视图定义语句格式

```
CREATE  VIEW
        <视图名>  [(<列名>  [，<列名>]...)]
      AS  <子查询>
      [WITH  CHECK  OPTION];
```

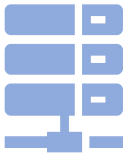


视图

- 视图是一种虚表（virtual table），是从一个或几个基本表（或视图）导出的表
  - 视图其实是在数据字典中存储的一条Select 语句

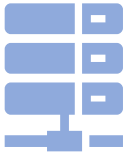
[例17] 建立信息系学生的视图

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



视图

- 视图是一种虚表（virtual table），是从一个或几个基本表（或视图）导出的表
  - 视图其实是在数据库中存储的一条Select 语句
  - DBMS执行CREATE VIEW语句时只是把视图的定义存入数据字典，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
  - 对于某些视图（可更新视图），可以对视图执行数据更新操作，数据库会根据视图定义去更新对应的基本表数据



视图

■ 视图定义

在定义视图时，组成视图的属性列名需全部省略或全部指定

- 当如下情形时，全部省略：

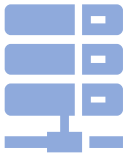
  视图属性由子查询中SELECT目标列中的诸字段组成

- 当如下情形时，需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

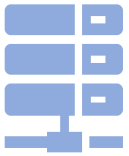


视图

■ 视图定义

[例18] 建立信息系选修了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  Sdept= 'IS' AND
        Student.Sno=SC.Sno AND
        SC.Cno= '1';
```

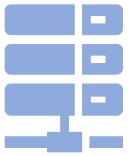


视图

- 视图定义
  - 可以基于视图建立新的视图

[例19] 建立信息系选修了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;
```



视图

■ 视图定义

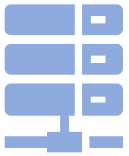
● 带表达式的视图

[例20] 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，2000-Sage
FROM  Student
```

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



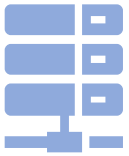
视图

- 视图定义
  - 带分组的视图

[例21] 将学生的学号及他的平均成绩定义为一个视图

假设SC表中“成绩”列Grade为数字型

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



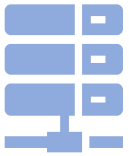
视图

- 视图定义
  - 不建议以 SELECT * 方式创建视图，因这样创建的视图可扩展性差

[例22]将Student表中所有女生记录定义为一个视图

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

缺点：当修改基表Student的结构后，Student表与F_Student1视图的映象关系会 被破坏，导致该视图不能正确工作。



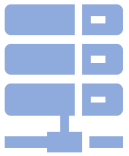
视图

- 视图定义
  - 不建议以 SELECT * 方式创建视图，因这样创建的视图可扩展性差

[例23]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW
        F_Student2 (stdnum，name，sex，age，dept)
AS  SELECT Sno，Sname，Ssex，Sage，Sdept
    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。



视图

■ 视图使用

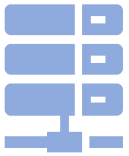
- 视图查询：对视图的查询与对基本表的查询完全相同
- 视图更新
  - ✓ 从用户角度，对视图的更新与对基本表的更新方法上也完全相同
  - ✓ 但是存在一些不可更新视图，因为对这些视图的更新不能唯一地/有意义地转换成对相应基本表的更新

如：下述视图S_G为不可更新视图

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

下述更新语句无法转换为对基本表的更新

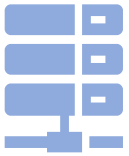
```
UPDATE  S_G
SET       Gavg=90
WHERE  Sno= '95001'；
```



视图

■ 视图使用

- 一般来说，允许对行列子集视图进行更新
- 其他情况，参见ISO对于可更新视图的规则要求：
  - DISTINCT is not specified.
  - Every element in SELECT list of defining query is a column name and no column appears more than once.
  - FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.
  - No nested SELECT referencing outer table.
  - No GROUP BY or HAVING clause.
  - Also, every row added through view must not violate integrity constraints of base table.
- 不同数据库对于视图是否可更新也有自己的一些规定



视图

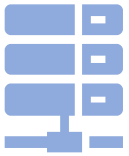
这个视图是否可更新?

```
CREATE VIEW MyStudent (num, name) AS
SELECT sNumber, sName FROM Student
WHERE sNumber NOT IN (SELECT sNumber FROM
Student);
```

理论上可更新

但很多数据库不给更新

假设可更新，该视图更新后会有什么现象?



视图

■ With Check Option

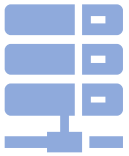
[例24] 建立信息系学生的视图。

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

通过该视图插入下述记录（95008，‘韩萍’，20，‘CS’）：

```
Insert into IS_Student values(95008，’韩萍‘，
20，’ CS’）
```

通过视图查看学生信息，出现什么问题?

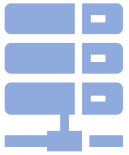


视图

- With Check Option
  - 透过视图进行增删改操作时，不得破坏视图定义中的谓词条件（即子查询中的条件表达式）

[例25] 建立信息系学生的视图。

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

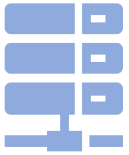


视图

- 视图的作用
  - 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，如对于下述的情形：

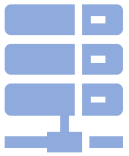
    - ✓ 基于多张表连接形成的视图
    - ✓ 基于复杂嵌套查询的视图
    - ✓ 含导出属性的视图



视图

- 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

视图对基本表进行数据抽取和组合，形成不同的观察窗口，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

- 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

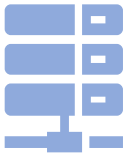
例：数据库逻辑结构发生改变

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)



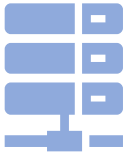
视图

- 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

通过建立一个视图Student：

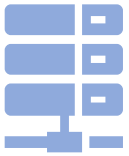
```
CREATE VIEW  Student(Sno，Sname，Ssex，Sage，Sdept)
AS
SELECT  SX.Sno，SX.Sname，SY.Ssex，SX.Sage，SY.Sdept
FROM  SX，SY
WHERE  SX.Sno=SY.Sno;
```

使用户的外模式保持不变，从而对原Student表的查询程序不必修改



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
    - ✓ 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
    - ✓ 通过WITH CHECK OPTION对关键数据施加操作限制



视图

■ 视图的作用

● 视图能够对机密数据提供安全保护

[例26] 建立1号课程的选课视图，并要求透过该视图进行的更新操作只涉及1号课程，同时对该视图的任何操作只能在工作时间进行。

```
CREATE VIEW IS_SC
        AS
SELECT Sno，Cno，Grade
FROM  SC
WHERE  Cno= '1'
AND  TO_CHAR(SYSDATE,'HH24')  BETWEEN  9  AND  17
AND  TO_CHAR(SYSDATE,'D')  BETWEEN  2  AND  6
WITH CHECK OPTION;
```